AUTOMATED ISOTOPE IDENTIFICATION AND QUANTIFICATION USING ARTIFICIAL NEURAL NETWORKS

BY

MARK KAMUDA

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Nuclear, Plasma, and Radiological Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Doctoral Committee:

Assistant Professor Kathryn Huff, Chair
Professor Mark Hasagawa-Johnson
Associate Professor Tomasz Kozlowski
Adjunct Research Assistant Professor Clair Sullivan
Professor Rizwan Uddin

ABSTRACT

Current radioisotope identification devices struggle to identify and quantify isotopes in low-resolution gamma-ray spectra in a wide range of realistic conditions. Trained gamma-ray spectroscopists typically rely on intuition when identifying isotopes in spectra. A trained gamma-ray spectroscopist can inject their intuition into pattern recognition algorithms by creating training datasets and intelligently choosing a machine learning model for a task. Algorithms based on feature extraction such as peak finding or ROI algorithms work well for well-calibrated high resolution detectors. For low-resolution detectors, it may be more beneficial to use algorithms that incorporate more abstract features of the spectrum. To investigate this, we simulated datasets and used them to train artificial neural networks (ANNs) for identification and quantification tasks using gamma-ray spectra. Because the datasets were simulated, this method can be extended to a variety of gamma-ray spectroscopy tasks. Models we investigated include dense, convolutional, and autoencoder ANNs. In this work we introduce `annsa`, an open source Python package capable of creating gamma-ray spectroscopy training datasets and applying machine learning models to solve spectroscopic tasks. Using `annsa`, we found that identification performance in simulated spectra was sensitive to the source-to-background ratio, detector gain setting, and shielding. Performance was less sensitive to the source-detector height and detector resolution. We demonstrate `annsa`'s capabilities on a source interdiction classification problem, outperforming a peak-based Bayesian classifier for source identification. We also demonstrate `annsa` on a uranium enrichment quantification problem which shows an accuracy useful for homeland security applications.

ACKNOWLEDGMENTS

I would like to first thank my advisor, Dr. Kathryn Huff, for her guidance and support during my graduate studies. I also want to thank my masters advisor, Dr. Clair Sullivan, for inspiring me to begin my graduate studies and reach further. I owe Dr. Jacob Stinnett for his initial research suggestion and discussions that started this work.

I also want to thank my friends and loved ones for their support, coffee breaks, and happy hour conversations that kept me sane.

This work was funded by the Consortium for Verification Technology under Department of energy National Nuclear Security Administration award number DE-NA0002534.

TABLE OF CONTENTS

LIST OF TABLES vi

LIST OF FIGURES viii

CHAPTER 1 INTRODUCTION 1
1.1 Introduction and Motivation 1
1.2 Neural Network History 3
1.3 Gamma-Ray Spectroscopy for Isotope Identification and Quantification 4
1.4 Automated Isotope Identification Using ANNs 6
1.5 Chapter Conclusion 7

CHAPTER 2 THEORY 8
2.1 Gamma-ray Spectroscopy 8
2.2 Machine Learning 20
2.3 annsa 40
2.4 Chapter Conclusions 52

CHAPTER 3 MACHINE LEARNING MODELS EXPLORED 53
3.1 Training Templates Overview 53
3.2 Hyperparameter Search 57
3.3 Datasets Used for the Hyperparameter Search 58
3.4 Hyperparameter Search Results 60

CHAPTER 4 URBAN SOURCE IDENTIFICATION RESULTS AND DISCUSSION 72
4.1 Learning Curve Analysis 72
4.2 Performance in Simulated Datasets 75
4.3 Performance Identifying Measured Spectra 88
4.4 Comparison with Peak-Based Bayesian Classifier 104
4.5 Chapter Discussion and Conclusion 106

CHAPTER 5 URANIUM ENRICHMENT REGRESSION RESULTS AND DISCUSSION 108
5.1 Uranium Enrichment Measurement Background 108

5.2 Problem Description and Training Dataset Overview 110
5.3 Results - Simulated Data 113
5.4 Results - Measured Spectra 117
5.5 Discussion and Conclusion 121

CHAPTER 6 CONCLUSIONS AND FUTURE WORK 122
6.1 General Conclusions 122
6.2 Suggested Future Work 124

REFERENCES 126

LIST OF TABLES

2.1 Typical energy resolutions of common gamma-ray detector types. Reproduced from [1]. 14
2.2 `annsa`'s DNN parameters. 46
2.3 `annsa`'s 1D CNN parameters. 48
2.4 `annsa`'s DAE parameters. 50

3.1 Fixed GADRAS-DRF template simulation parameters. 57
3.2 Default data augmentation parameters. 57
3.3 Range of parameters used for the simple dataset. 59
3.4 Range of parameters used for the complete dataset. 59
3.5 Range of hyperparameters explored for the DNN. 61
3.6 Optimum DNN hyperparameter combination found for the simple and complete dataset. 65
3.7 Range of hyperparameters explored for the CNN. 66
3.8 Optimum CNN hyperparameter combination found for the simple and complete dataset. 70

4.1 Normalized area under the learning curves generated using the simple training dataset. 74
4.2 Normalized area under the learning curves generated using the complete training dataset. 75
4.3 Default parameters used for all testing datasets. 77
4.4 ANN predictions of ^{137}Cs spectra measured with a relative gain shift of 0.59. Spectra were measured with a signal-to-background ratio of 1.0 for 60 s. Simple and complete ANNs are indicated by the S- and C- prefix respectively. Incorrect alarms are bolded. 93
4.5 F1 scores from the Bayesian classifier and ANNs. The highest F1 scores are shown in bold. 105
4.6 Isotopes incorrectly classified by the ANNs. Simple and complete ANNs are indicated by the S- and C- prefix respectively. Entries with dashes indicate a correct identification. 106

5.1 Specific activities for 50 year old uranium isotopes and their ingrown daughters. 112

5.2 Range of parameters used for the uranium enrichment dataset. 113
5.3 Default parameters used for all generalization datasets. 114
5.4 Uranium sample description. 118

LIST OF FIGURES

2.1 Energy dependence for gamma-ray interactions in NaI(Tl). Reproduced from [2]. 9
2.2 Diagram of a Compton scattering event. Reproduced from [2]. 10
2.3 Diagram of energy absorbed in an idealized Compton scattering event. The solid and dashed line represent idealized spectra with and without including electron binding energy. Reproduced from [3]. 11
2.4 Diagram of pair production. Reproduced from [3]. 12
2.5 Theoretical spectrum from pair production. Reproduced from [2]. 13
2.6 ^{133}Ba spectrum measured using common detector materials. Reproduced from [1]. 14
2.7 Monte Carlo simulations of background component spectra in NaI(Tl). Reproduced from [4]. 15
2.8 Map of uranium concentrations in the United States. Reproduced from [5]. 16
2.9 Map of thorium concentrations in the United States. Reproduced from [5]. 17
2.10 Map of potassium concentrations in the United States. Reproduced from [5]. 17
2.11 Temperature vs relative photopeak position for an ORTEC Model 905-3 2x2 NaI(Tl) detector. Reproduced from [6]. . . . 18
2.12 The energy resolution of 662 keV γ-rays measured with a 25mm x 30mm NaI(Tl) crystal using 2 μs and 12 μs peaking times. Reproduced from [7]. 19
2.13 Example ANN with input neurons A_n, hidden neurons B_j, and output neurons C_k [8]. 21
2.14 Summary of the operation of a single neuron [8]. 21
2.15 Example training and validation error curves. 24
2.16 An example of k-folds cross validation. 25
2.17 Example of a single-layer neural network with two inputs (x_1 and x_2), three classes (y_1, y_2, y_3), and a bias neuron set to one. 26

2.18 A dataset describing a three class function. Each class is represented by a different color. 27
2.19 Example training paths for a large learning rate and a small learning rate. 30
2.20 Two examples of data augmentation using an image of a cat. The image to the left is the original. The top right image is augmented using a horizontal flip. The bottom right image is augmented using blur. 35
2.21 A comparison between a grid search and a random search for hyperparameter optimization. In this example, performance is strongly tied to one hyperparameter. The green function represents the effect of an important hyperparameter on a cost function while the yellow function represents the effect for an unimportant hyperparameter. Figure reproduced from [42]. 36
2.22 Architecture of LeNet-5, a CNN created for digit recognition. Image reproduced from [9] 38
2.23 An example of an autoencoder. Image reproduced from [10]. . 39
2.24 The main modules in the `annsa` package. 40
2.25 Example csv entries for the source and background template datasets. 41
2.26 An example of the procedure used to normalize a template spectrum. 43
2.27 An example of the template sampling process combining ^{60}Co and background template spectra into a dataset entry. . . 44
2.28 `annsa`'s DNN structure. 45
2.29 The 1D convolutional-pooling operation included in `annsa`. . . 47
2.30 `annsa`'s CNN structure. 47
2.31 `annsa` DAE structure. This example demonstrates a background subtracting denoising autoencoder. 49
2.32 An example of a DAE's encoding being loaded into a classification DNN. 49
2.33 `annsa`'s CAE structure. This example demonstrates a background subtracting denoising autoencoder. 51
2.34 An example of a CAE's encoding being loaded into a CNN. . . 51
2.35 Diagram of AWS use. 52

3.1 GADRAS-DRF GUI showing parameters used to simulate the Ortec 905-3 2x2-in NaI(Tl) detector used in this work. . . 54
3.2 GADRAS-DRF measurement diagram. Environmental scatter is approximated using the photon scatter terms from Figure 3.1. 55
3.3 Comparison of a ^{60}Co spectrum simulated using various source-detector distances. 55

3.4 Comparison of a ^{60}Co spectrum simulated using various source-detector heights off the ground. 56
3.5 Comparison of a ^{60}Co spectrum simulated with various FWHM parameters. 56
3.6 Hyperparameter search workflow. 58
3.7 Hyperparameter search workflow. 60
3.8 Random hyperparameter search efficiency curves for the DNN using the simple dataset. Red crosses indicate individual experiments. 62
3.9 Random hyperparameter search efficiency curves for the DNN using the complete dataset. Red crosses indicate individual experiments. 62
3.10 Effect of dense hyperparameters on the validation dataset's final F1 score. 64
3.11 Random hyperparameter search efficiency curves for the CNN using the simple dataset. Red crosses indicate individual experiments. 67
3.12 Random hyperparameter search efficiency curves for the CNN using the complete dataset. Red crosses indicate individual experiments. 67
3.13 Effect of CNN hyperparameters on the validation dataset's final F1 score. 69
3.13 Effect of CNN hyperparameters on the validation dataset's final F1 score. 70

4.1 Training process used to train models for the learning curves. . 73
4.2 Learning curves for each simple model. 74
4.3 Learning curves for each complete model. 75
4.4 Model averaging process used to evaluate ANNs using testing datasets. 76
4.5 F1 score dependence on source-detector height measured using simulated data. 78
4.6 Precision dependence on source-detector height measured using simulated data. 78
4.7 Recall dependence on source-detector height measured using simulated data. 79
4.8 F1 score dependence on source-detector distance measured using simulated data. 80
4.9 Precision dependence on source-detector distance measured using simulated data. 80
4.10 Recall dependence on source-detector distance measured using simulated data. 81
4.11 F1 score dependence on detector resolution measured using simulated data. 82

4.12 Precision dependence on detector resolution measured using simulated data. 83
4.13 Recall dependence on detector resolution measured using simulated data. 83
4.14 F1 score dependence on shielding measured using simulated data. 84
4.15 Precision dependence on shielding measured using simulated data. 85
4.16 Recall dependence on shielding measured using simulated data. 85
4.17 F1 score dependence on gain using simulated data. 87
4.18 Precision dependence on gain using simulated data. 87
4.19 Recall dependence on gain using simulated data. 88
4.20 Diagram of the laboratory setup used to measure radioactive sources. 89
4.21 Example ^{137}Cs measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively. 91
4.22 Effect of calibration gain on the posterior probability of ^{137}Cs measured using real ^{137}Cs spectra. 92
4.23 Example ^{60}Co spectra measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively. 93
4.24 Effect of calibration gain on the posterior probability of ^{60}Co measured using real ^{60}Co spectra. 94
4.25 Example ^{133}Ba spectra measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively. 95
4.26 Effect of calibration gain on the posterior probability of ^{133}Ba measured using real ^{133}Ba spectra. 96
4.27 Example ^{152}Eu spectra measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively. 97
4.28 Effect of calibration gain on the posterior probability of ^{152}Eu measured using real ^{152}Eu spectra. 97
4.29 Spectra of ^{137}Cs shielded with increasing amounts of iron measured for 60 s. 98
4.30 Effect of iron shielding thickness on the posterior probability of ^{137}Cs measured using real shielded ^{137}Cs spectra. 99
4.31 Spectra of ^{60}Co shielded with increasing amounts of iron measured for 60 s. 99
4.32 Effect of iron shielding thickness on the posterior probability of ^{60}Co measured using real shielded ^{60}Co spectra. 100

4.33 Effect of aluminum shielding thickness on the posterior probability of ^{133}Ba measured using real shielded ^{133}Ba spectra. 101
4.34 Shielding generalization performance in real ^{133}Ba spectra. . . 102
4.35 Spectra of ^{152}Eu shielded with increasing amounts of iron measured for 60 s. 103
4.36 Effect of iron shielding thickness on the posterior probability of ^{152}Eu measured using real shielded ^{152}Eu spectra. 103

5.1 27% enriched uranium spectrum measured with a NaI(Tl) detector. 109
5.2 Training and prediction averaging. 111
5.3 Diagram of MCNP simulation (not to scale). 112
5.4 Simulated 93% enriched uranium spectra various amounts of shielding. 114
5.5 Each model's prediction for the uranium enrichment of spectra simulated with different shielding thicknesses. Shielding conditions are indicated below each figure. 115
5.6 Each model's prediction for the uranium enrichment of simulated spectra calibrated with different gain settings. The magnitude of the applied relative gain shift are shown below each figure. 116
5.7 Simulated 93% enriched uranium spectra with three different relative gain settings. Energy calibration shown is based on the 1.0 relative gain setting. 117
5.8 Simulated 10% enriched uranium spectra with three different relative gain settings. Energy calibration shown is based on the 1.0 relative gain setting. 117
5.9 Enriched uranium spectra recalibrated to the 186 keV and 1001 keV photopeaks. 118
5.10 Each model's predicted uranium enrichment for measured uranium spectra. 119
5.11 Each model's prediction for the uranium enrichment of measured spectra calibrated with different gain settings. The magnitude of the applied relative gain shift are shown below each figure. 120

CHAPTER 1

INTRODUCTION

1.1 Introduction and Motivation

Gamma-ray spectroscopy plays an important role in homeland security and nonproliferation technologies. By analyzing the gamma-ray spectrum emitted by an object, naturally occurring radioactive material (NORM) can be distinguished from threat isotopes potentially contained by the object. Threat isotopes include undeclared industrial, medical, and special nuclear material (SNM) sources that can be used in a dirty bomb or nuclear explosive device. Detecting these materials is the first step to intercepting them and preventing nuclear material proliferation.

Machine learning algorithms have not been rigorously explored for the analysis of gamma-ray spectra. To provide the gamma-ray spectroscopy community with a tool to create training datasets and train machine learning models for spectroscopic tasks, we have created the open source Python package `annsa` (artificial neural networks for spectroscopic analysis). In this work we apply `annsa` to two problems: source interdiction classification and uranium enrichment quantification.

Common detection materials for commercial radioisotope identification devices (RIID) are the low-resolution sodium iodide (NaI(Tl)) [11], medium-resolution cadmium zinc telluride (CZT), and high-resolution high purity germanium (HPGe). All of these materials have sufficient resolution to perform gamma-ray spectroscopy. While medium- and high-resolution materials offer better performance, they suffer from several drawbacks. These materials cost more and are difficult to manufacture in large volumes. Larger detector volumes have an increased absolute efficiency, reducing measurement times. In addition to these drawbacks, HPGe detectors require cryogenic cooling. This greatly increases the weight and cost of portable HPGe detectors. De-

spite the reduced resolution, NaI(Tl) is standard in the RIID industry due to its low cost, ability to be manufactured in large volumes, high intrinsic efficiency. This dissertation specifically uses the Ortec 905-3 2x2-in. NaI(Tl) cylindrical scintillation detector.

Reported commercial RIID performance in automated isotope identification is generally poor [12, 13, 14]. One study found that seven commercial RIIDs had an average of less than 50% correct identifications for SNM, industrial, and medical sources [13]. Another study found that a 2 mm stainless steel plate (a moderate amount of shielding) was enough to eliminate correct identifications in four commercially available RIIDs [12]. Because of the importance of an alarm - and because of inadequate RIID performance in automated isotope identification - the U.S. Department of Energy (DOE) employs a team of on-call spectroscopists to resolve RIID alarms [15]. Because of their importance, RIID detection systems need improvement.

RIID improvements fall into two categories: improvements in the quality of the spectrum and improvements in the identification algorithms [16]. Spectral quality is largely determined by the energy resolution of the radiation detection material. Improvements in energy resolution often increase manufacturing cost and reduce feasible detector size. Active research into advanced detection materials has not yet produced a material economically competitive with NaI(Tl). Because of this, RIID improvements should focus on the identification algorithms using this industry standard material [13].

Despite widespread use, NaI(Tl) detectors have several issues that complicate automated identification. The first issue is the low resolution of NaI(Tl). The poor resolution makes some photopeaks unresolvable from each other, complicating identification.

The second issue is calibration drift due to changes in environmental temperature and voltage drifts in the photomultiplier tube due to changes in count rate [2, 3]. This drift affects the locations and shapes of features in a gamma-ray spectrum. To solve this, the detector must recalibrate to a reference source (possibly built-in) or a naturally occurring background source. These methods fall short for different reasons. Built-in sources need to be periodically replaced, add an unwanted signal to a spectrum, and complicate commercial shipment. Recalibration sources can be included separately, requiring the user find this source and periodically recalibrate - complicating practical use. Calibrating from a background reference (typically the

1.460 MeV gamma-ray peak from ^{40}K or the 2.614 MeV gamma-ray peak from the thorium decay series) requires a significant measurement time for a useful signal-to-noise ratio. Short measurement times common in homeland security measurements make this option infeasible.

The third issue is a non-linear energy response which manifests as a small second order term in the detector's calibration. While not specific to NaI(Tl) detectors, the fourth issue affecting identification algorithms is background radiation from naturally occurring radioactive material (NORM). NORM radiation can change both in intensity and composition based on location and weather. This background signal can obscure features in gamma-ray spectra.

Despite the drawbacks outlined above, our previous published work has demonstrated that ANNs are capable of identifying multiple isotopes in unknown backgrounds with a wide range of calibration settings [17, 8, 18]. Furthermore, the work presented in this dissertation demonstrates that ANNs are capable of performing both classification and quantification tasks using low-resolution gamma-ray spectra.

1.2 Neural Network History

Scientists first theorized artificial neural networks in the 1940s as a model of how complex biological systems like neuron bundles learn and remember [19, 20]. These theories hypothesized that learning takes place by reinforcing neural connections corresponding to some beneficial behavior. The first implementation of an ANN came in 1958 in the form of a machine named The Mark I Perceptron [21, 22]. The Mark I Perceptron was a physical neural network whose weights were changed with motor-controlled potentiometers. The Mark I Perceptron implemented a two-class image recognition problem using a 400-pixel camera. In this context, a class represents an item in a set of mutually exclusive items (e.g. dog/cat or on/off). The device also included an algorithm (the perceptron learning algorithm) to learn optimum weights for a given task. Unfortunately, the single-layer perceptron failed in cases without linearly separable classes in the data because it was based on linear combinations of fixed basis functions [23, 24]. This realization and other perceived flaws led to a temporary decline in neural network research.

Further advances were made when Rumelhart et al. formalized error backpropagation and its application to training ANNs [25]. Error backpropogation was found to be a simple and powerful method to update an ANN to learn arbitrary functions. Combining error backpropogation with gradient descent allowed for efficient ANN training. Algorithms based on the backpropogation of error are now the most common method to train ANNs.

Currently, ANNs can solve many, diverse problems. ANNs have shown promise in everyday problems such as handwritten zip code recognition [26], image recognition [27], and fingerprint identification [28]; as well as more complicated problems such as lung cancer classification based on MRI images [29], estimating surface soil moisture from high-resolution aerial images of cropland [30], and stock market forecasting [31].

1.3 Gamma-Ray Spectroscopy for Isotope Identification and Quantification

Rawool-Sullivan et al. identified a common workflow performed by a group of trained gamma-ray spectroscopists [32]. This workflow includes discriminating source photopeaks from the background signal, adjusting the calibration using background photopeaks, and checking for shielding effects in the low-energy photopeaks. Once the spectroscopist identifies photopeaks, they use prior knowledge or consult a database of isotope decay energies to identify isotopes in the spectrum. The researchers noted that spectroscopists often employ a mixture of factual knowledge and intuition developed from analyzing hundreds of gamma-ray spectra in their analysis. The researchers also noted the difficulty in incorporating this subjective analysis into an automated algorithm.

Many automated radioisotope identification methods are available, but few perform well given a low-resolution gamma-ray spectrum of a mixture of radioisotopes. Automated RIID methods explored in research include library comparison algorithms, region of interest (ROI) algorithms, principle component analysis (PCA), and template matching. While these algorithms may work well in laboratory settings, they often offer unacceptable performance in more realistic conditions.

Library comparison algorithms attempt to match photopeak energies found

in a gamma-ray spectrum with those found in a library of known isotope decay energies. Drifts and uncertainties in detector calibration can lead to misidentifying photopeaks, leading to incorrect isotope identifications [15]. To automate this method, a separate algorithm is required to extract photopeak centroids in the presence of calibration drift and an unknown background signal. While research on methods for photopeak extraction are ongoing [33, 34, 35], they face difficulties when a large number of photopeaks overlap in a spectrum [36], such as when a low-resolution detector measures a mixture of radio-isotopes.

ROI algorithms define regions in a spectrum where they expect target radioisotope photopeaks. These algorithms then compare counts in these regions to a measured or expected background. Significant elevation in counts in a target isotopes ROIs indicates the presence of that isotope. Similar to library comparison algorithms, ROI algorithms operate poorly when photopeaks of different radioisotopes overlap [15]. Because of this, large isotope libraries perform poorly using this method. Similarly to the library comparison algorithm, calibration drift may shift photopeaks into neighboring ROIs, leading to incorrect identification. Despite drawbacks, an ROI method has been used to differentiate normally occurring radioactive material (NORM) from special nuclear material (SNM) using plastic scintillators [37].

PCA can also be applied to radioisotope identification. PCA aims to reduce a dataset's dimensionality into uncorrelated variables [38]. Using a subset of these principle components, the data may be represented in a reduced space of orthogonal bases that contains most of the information present in the original data. Isotopes in the the transformed data can then be clustered using methods like K-means, Mahalanobis distance, or k-nearest neighbors [39, 40]. PCA has been applied to isotope identification using plastic scintillators [41] and anomaly detection using both plastic scintillators and NaI(Tl) detectors [42]. Despite PCA's progress in some isotope identification problems, application using PCA to separating isotope mixtures in gamma-ray spectra could not be found.

Template matching algorithms find an example in a database of gamma-ray spectra that most closely matches a measured spectrum [15]. The spectral database can contain multiple detector calibration settings, shielding materials, and source-to-detector distances. Quality of fit can be measured using a hypothesis test such as chi-squared test or correlation coefficient. While a suf-

ficiently large spectral database can theoretically be used to identify almost any measured spectrum, the drawback of this method is the time necessary to compare a measured spectrum to the library and the computer memory necessary to store said library. Despite the drawbacks, researchers have made progress applying a multiple linear regression procedure to identify mixtures of isotopes using template matching [43].

The present work is motivated by the notion that incorporating the intuition identified by Rawool-Sullivan et al. can improve these algorithms. By carefully creating a training set of spectra and intelligently applying machine learning algorithms, a model can be trained that incorporates a spectroscopists intuition.

1.4 Automated Isotope Identification Using ANNs

A number of published papers apply ANNs to automated isotope identification. ANNs have been applied to peak fitting [44], isotope identification [45, 46], and activity estimation [45, 47]. Many of these publications rely on ROI methods [48], feature extraction [49], high-resolution gamma-ray spectra [50], and small libraries of isotopes. In addition, many assume perfect knowledge of detector calibration that is both linear and static. ANN training methods created for high-resolution gamma-ray spectra may not perform well when trained using low-resolution spectra. Because of the large discrepancy in resolution, spectral features exploited by a ANN trained on high-resolution spectra would be different than an ANN trained on low-resolution spectra. In addition, ANN training that relies on ROI methods may not perform well when ROIs overlap significantly (as previously explained). Feature extraction and ROI methods may also falter when the background radiation field is unknown or the detector calibration is unreliable.

Instead of training an ANN using predetermined ROIs or feature extraction, the present work hypothesizes that it is better to train the ANN with an entire gamma-ray spectrum. Due to perceived training issues and computational requirements associated with using the entire spectrum [48, 50], previous work in the gamma-ray spectroscopy community have avoided this approach. However, we have shown that training an ANN using the full spectrum can viably identify and quantify isotopes in gamma-ray spectra

[8, 17, 18]. Evidence in the present work also demonstrates that this method can overcome common gamma-ray spectroscopy issues like calibration shifts and identifying isotopes in spectra without clear spectral features.

Using machine learning for isotope identification and quantification offers many advantages over methods previously explained. The problem of determining the feature extraction technique and optimal algorithm is avoided by training an algorithm to identify the important features of a gamma-ray spectrum and simultaneously perform identification or quantification. Using a machine learning approach also has an advantage in the flexibility of its training set and learning objective. Because this method uses simulated gamma-ray spectra to train the model, it does not restrict the number of isotopes allowed in the library. This allows us to cheaply generate the ANN training set using mixtures of exotic, dangerous, or short-lived isotopes that are not easily accessible. Because the training set and learning objective of a machine learning algorithm are flexible, we can train similar deep learning models to perform diverse tasks like source interdiction or uranium enrichment measurements.

1.5 Chapter Conclusion

This section outlined the motivation for applying machine learning algorithms to gamma-ray spectroscopy tasks. In subsequent chapters we describe how we implemented machine learning algorithms to solve these problems. Chapter 2 outlines the theory behind features in gamma-ray spectra, ANN operation, and the software package `annsa` (Artificial Neural Networks for Spectroscopic Analysis) [51] we created to couple gamma-ray transport software and machine learning models. Chapter 3 describes how we optimized machine learning models for gamma-ray spectroscopy. Chapters 4 and 5 demonstrate implementations of these models for performing source identification in an urban setting and uranium enrichment measurements, respectively. The performance of these models are benchmarked on a series of real and simulated gamma-ray spectra. Finally, in Chapter 6 we discuss this work's contribution to the field of automated gamma-ray spectroscopy and suggest future work.

CHAPTER 2

THEORY

This chapter covers how radiation detectors convert gamma-rays into spectroscopic signals, the theory behind machine learning with artificial neural networks, and the open source Python package, `annsa` (Artificial Neural Networks for Spectroscopic Analysis) [51], we created to couple gamma-ray transport software and machine learning algorithms.

2.1 Gamma-ray Spectroscopy

In this section, we describe the physical processes responsible for features in gamma-ray spectra and how some of these features complicate gamma-ray spectroscopy.

2.1.1 Energy Deposition Mechanisms

When a photon interacts with a radiation detection medium (e.g. a NaI(Tl) crystal), it deposits some or all of its energy into the material. There are three main methods of interaction: the photoelectric effect, Compton scattering, and pair production. These effects are energy dependent, as seen in Figure 2.1. After energy is deposited in the material, scintillation light produced by the crystal is collected and amplified by the detector's electronics. This amplified signal is sent to a computer where it is recorded in a gamma-ray spectrum. The detector's signal at low energies (below 30 keV) is dominated by noise. To remove this noise from the spectrum, a low level discriminator is often used to reject the signal below a certain number of channels. In this work, we apply a low level discriminator by setting the first 10 channels' counts in each spectrum to zero. This low level discriminator removes energies below 30 keV in a detector with 1024 channels calibrated

with a maximum energy of 3 MeV.

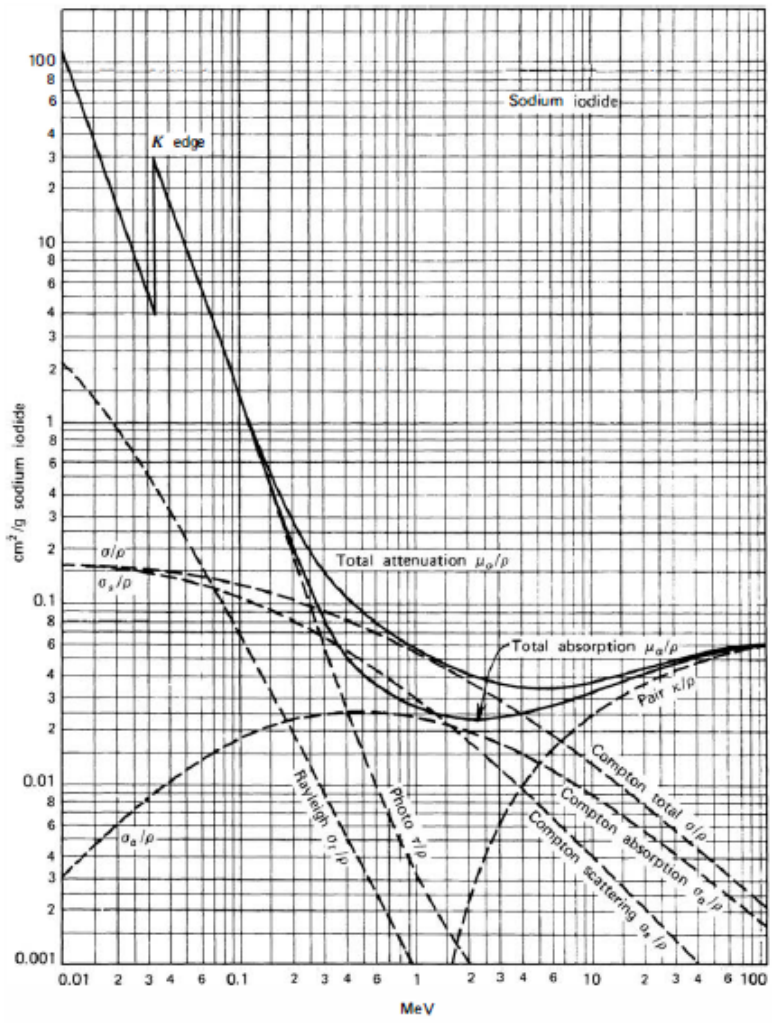


Figure 2.1: Energy dependence for gamma-ray interactions in NaI(Tl). Reproduced from [2].

If the photon's total energy is deposited in the detector, a count is recorded in the spectrum's photopeak. If the photon deposits only a part of its energy and escapes the detector, a count will be recorded somewhere below the full-energy photopeak. The following sections describe intrinsic mechanisms that add and remove counts from the full-energy peak.

2.1.1.1 Photoelectric Effect

The photoelectric effect is a process that can occur when a photon with energy, E_γ, greater than the binding energy of a bound electron, E_b, is absorbed

by an atom. This absorption produces a photoelectron, which has the energy of the photon minus the binding energy of the electron

$$\begin{aligned} E_{e^-} &= E_\gamma - E_b \\ &= h\nu - E_b \end{aligned} \tag{2.1}$$

where

$$\begin{aligned} h &= \text{Planks Constant} \quad [J \cdot s] \\ \nu &= \text{The Photon's Frequency} \quad \left[\frac{1}{\text{s}}\right] \end{aligned}$$

where $E_\gamma = h\nu$ is the incident photon's energy. The photoelectron is typically absorbed by the material, resulting in full-energy deposition. This mechanism is primarily responsible for producing photopeaks in gamma-ray spectra. This process also creates an electron vacancy in the original atom. This vacancy is filled by another electron in the material, releasing a characteristic x-ray or Auger electron. The characteristic x-ray can either be reabsorbed by the material or escape. Because of their low energies, Auger electrons are typically quickly reabsorbed.

2.1.1.2 Compton scattering

Compton scattering is the interaction of a gamma-ray photon with an electron in absorbing material. This interaction is illustrated in Figure 2.2.

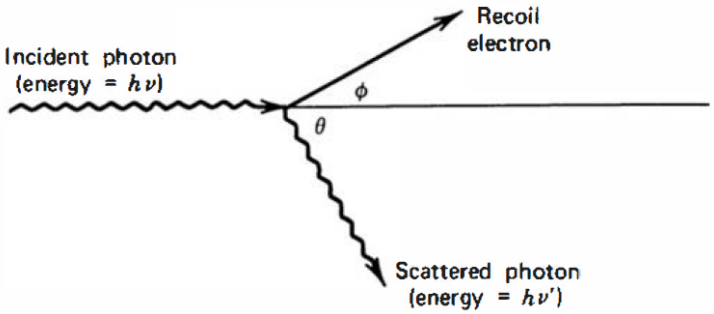


Figure 2.2: Diagram of a Compton scattering event. Reproduced from [2].

The energy of a photon scattered off a free electron,

$$E'_\gamma = \frac{E_\gamma}{1 + \frac{E_\gamma}{m_0 c^2}(1 - cos\theta)}, \tag{2.2}$$

depends on the scattering angle, θ, the incident photon's energy, E_γ, and the rest mass of the electron, $m_o c^2$. A photon scattering at an angle of 180^o deposits its maximum amount of energy into a medium. Photons that escape after this energy deposition create the Compton edge observed at $\theta = 180^o$, seen in the solid line in Figure 2.3. Photons that escape the detector after single or multiple Compton scatter events at angles less than 180^o create the continuum of energies known as the Compton continuum. Accounting for the binding energy of electrons in a medium produces a Compton continuum more similar to the dotted line in Figure 2.3.

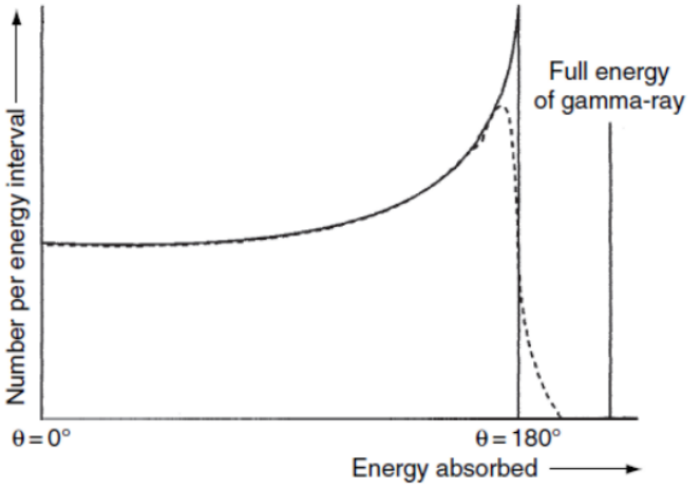


Figure 2.3: Diagram of energy absorbed in an idealized Compton scattering event. The solid and dashed line represent idealized spectra with and without including electron binding energy. Reproduced from [3].

The Compton continuum of higher-energy photons will add to lower-energy photopeaks, adding noise to and changing the baseline of lower-energy photons' signals. The geometry between the source and detector changes the Compton continuum's shape, making the Compton continuum's signal difficult to incorporate into an identification algorithm. This signal can be incorporated into template matching or ANN-based algorithms by including library or training set examples with various source-detector geometries.

2.1.1.3 Pair Production

When a photon with energy above 1022 keV (two times the rest mass energy of an electron) interacts with the Coulomb field of a nucleus, there is a probability that the photon will disappear and be replaced by an electron-positron pair. The positron will then annihilate with an electron in the medium, creating two 511 keV photons. This process is illustrated in Figure 2.4. If one or both of these annihilation photons escape the detector, they produce single or double escape peaks in a gamma-ray spectrum. As seen in Figure 2.5, single escape peaks occur at 511 keV below the full-energy peak and double escape peaks occur at 1022 keV below this peak. These 511 keV photons may also be measured by the detector, producing an annihilation radiation signal in the spectrum.

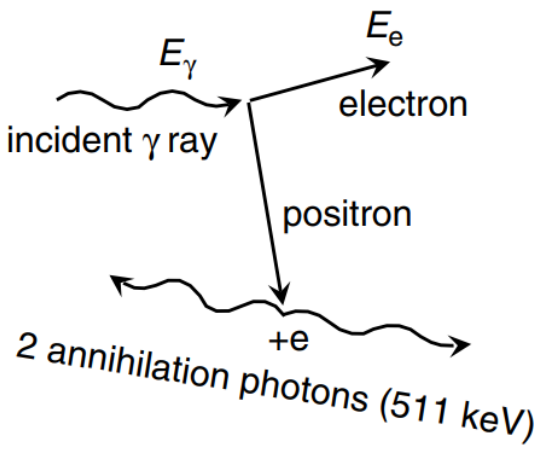


Figure 2.4: Diagram of pair production. Reproduced from [3].

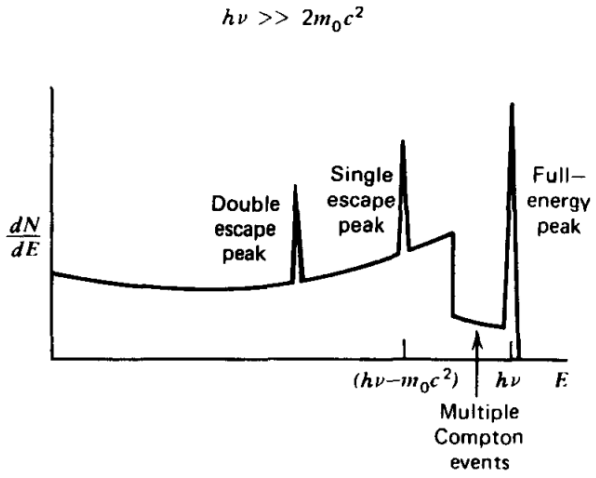


Figure 2.5: Theoretical spectrum from pair production. Reproduced from [2].

2.1.2 Energy Resolution

The energy resolution of a radiation detector determines the photopeak's distribution, specifically the full width of its Gaussian peak at half of its maximum value (FWHM), at a location in a spectrum. The FWHM is measured either in units of energy or as a percent of the peak's energy. This broadening is mostly due to statistical fluctuations in the number of information carriers produced in the detection system. Other factors that increase the resolution in scintillation detectors include nonproportionality of light yield per energy absorbed, the variance in photoelectron collection in the photocathode, and the variance in electrons produced in the photomultiplier tube [2]. Because semiconductor detectors more directly measure charge carriers, they do not suffer the resolution losses associated with transforming information carriers. The variance in information carriers produced for a given energy does not follow a Poisson process in semiconductor detectors. This effect is known as the Fano factor and it significantly decreases the resolution of semiconductor detectors below the theoretical Poisson limit [2]. Because the resolution changes as function of energy, the resolution of a 662 keV photon from ^{137}Cs is used as a standard for comparison. The energy resolution of three scintillator (NaI(Tl), LaBr, CeBr) and two semiconductor (CZT, HPGe) radiation detectors are compared in Table 2.1. Figure 2.6 compares a ^{133}Ba spectrum as measured by scintillation and semiconductor detectors. In this figure we

see that it is possible to resolve more photopeaks with higher-resolution detectors when compared to lower-resolution detectors.

Detector Type	**Full Width at Half Maximum (662 keV)**
NaI(Tl)	6 - 8 %
LaBr	2 - 4 %
CeBr	4 - 5 %
CZT	1 - 2 %
HPGe	<0.2 %

Table 2.1: Typical energy resolutions of common gamma-ray detector types. Reproduced from [1].

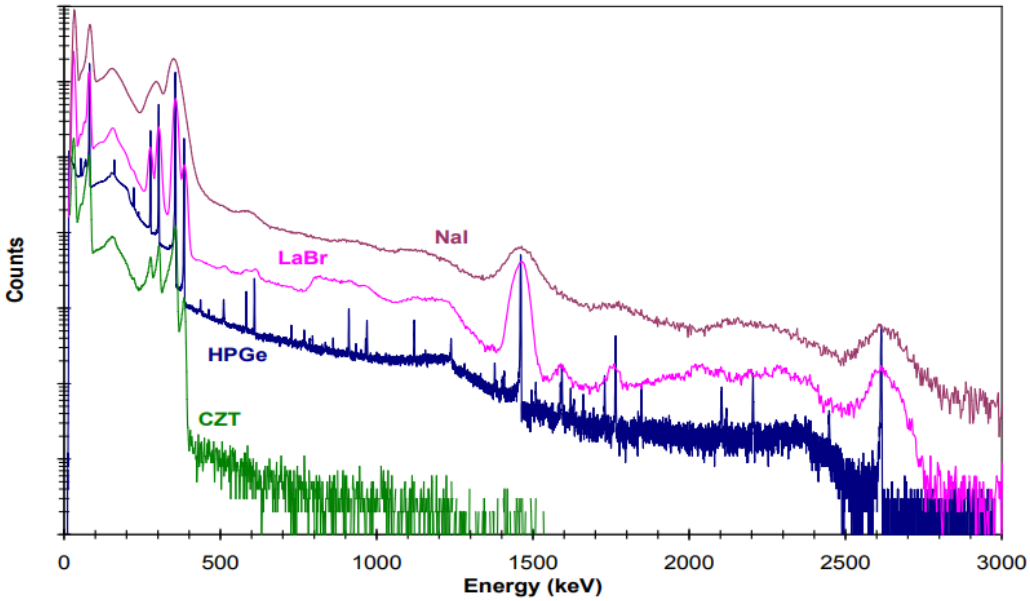


Figure 2.6: ^{133}Ba spectrum measured using common detector materials. Reproduced from [1].

Gamma-ray spectroscopy tasks in low resolution detectors, such as NaI(Tl), are complicated by overlapping photopeaks. Without well distinguished photopeaks, the signal from a source may be indistinguishable from noise or background. For example, in Figure 2.6 the two photopeaks around 600 keV are only clearly distinguishable in the high resolution HPGe detector. Wide photopeaks are also more likely to overlap with neighbors, potentially making them indistinguishable from each other, further complicating identification.

To address changes in resolution in this work, we simulated gamma-ray spectra with FWHM values of 7.0, 7.5, and 8.0. This range represents typical values for commercial NaI(Tl) detectors. Detectors can be fabricated with FWHM values smaller than this, but they are less common in commercial devices.

2.1.3 Background Radiation

A significant challenge to automated gamma-ray spectroscopy is the stochastic nature of the background radiation field. Spatial and temporal changes in background alter a gamma-ray spectrum's continuum and add a source of unwanted photopeaks. Background radiation comes from many sources including cosmic radiation and radioisotopes naturally distributed in soil and building materials. Isotopes in the soil primarily come from the uranium decay series, thorium decay series, and ^{40}K. The gamma-ray spectra from each of these sources are shown in Figure 2.7. These background signals span the range of energies useful to gamma-ray spectroscopy, adding multiple sources of independent noise and complicating analysis.

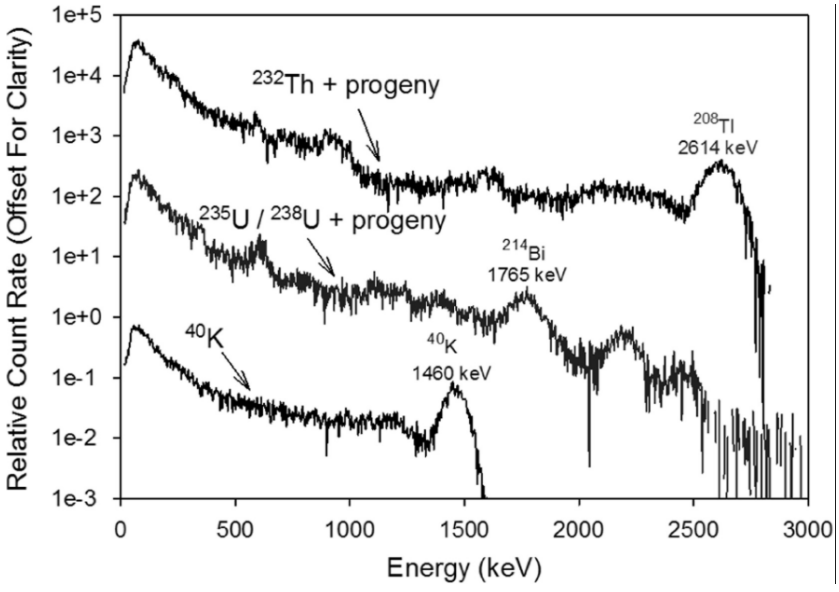


Figure 2.7: Monte Carlo simulations of background component spectra in NaI(Tl). Reproduced from [4].

These radiation components can change both spatially and temporally. Because subsurface rocks vary in their elemental composition, background

radiation isotope concentrations change geologically (Figures 2.8, 2.9, and 2.10). Local changes in soil composition and building materials also are significant enough to impact background. Because common building materials like concrete and granite contain radioactive elements, the proximity to these structures can also cause local variations in background.

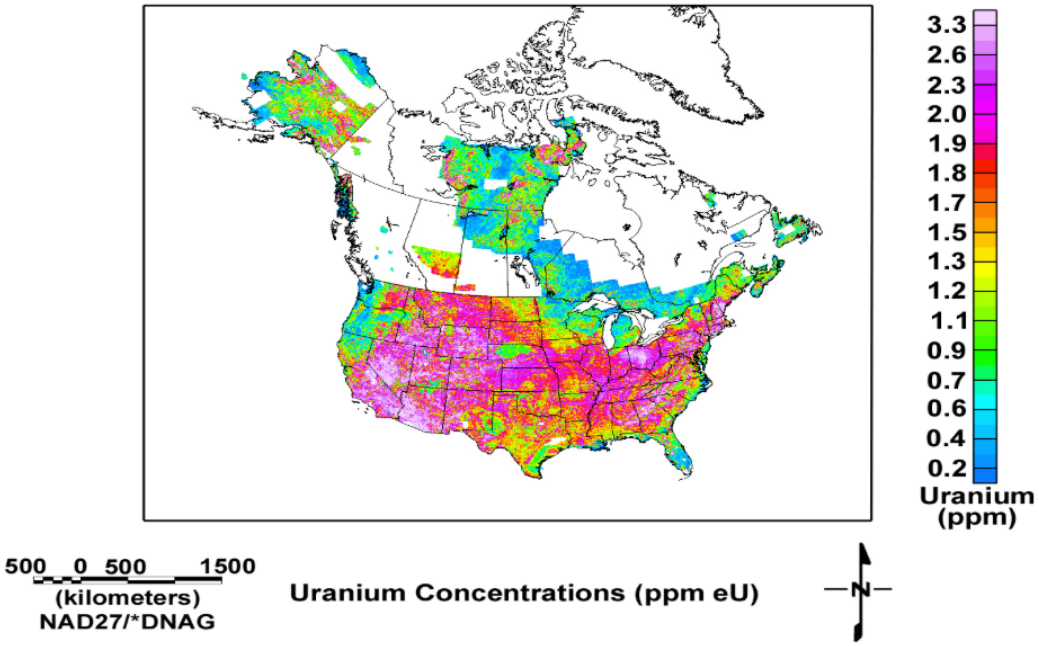


Figure 2.8: Map of uranium concentrations in the United States. Reproduced from [5].

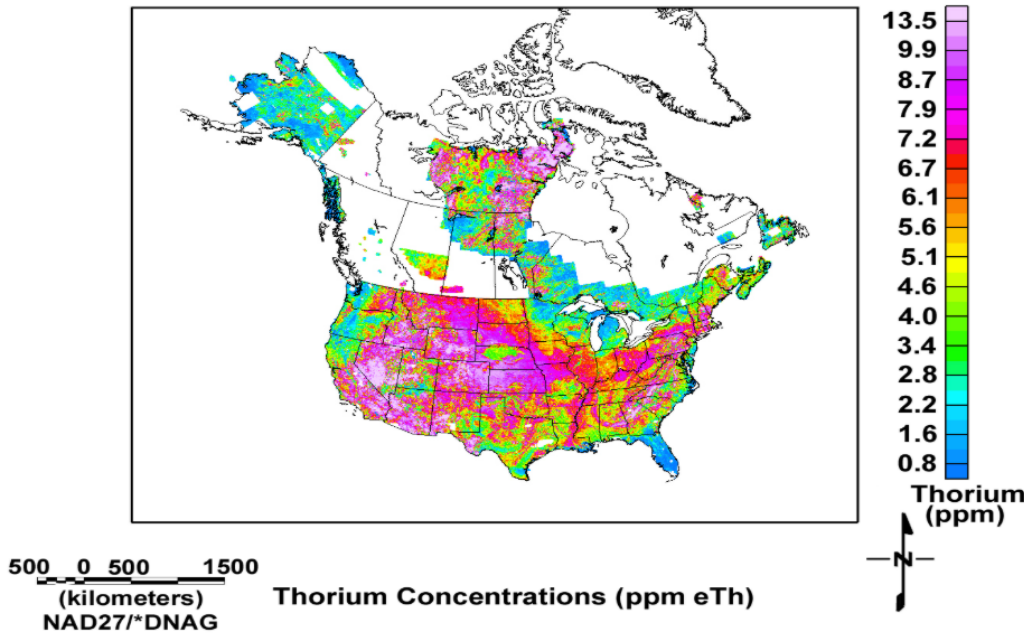


Figure 2.9: Map of thorium concentrations in the United States. Reproduced from [5].

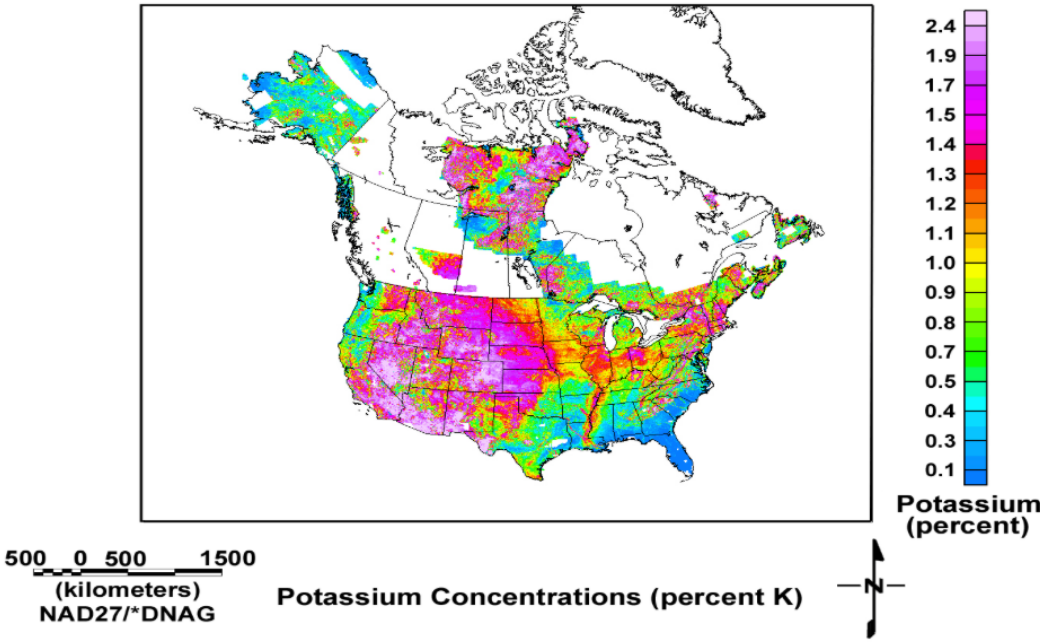


Figure 2.10: Map of potassium concentrations in the United States. Reproduced from [5].

Background also changes over time, largely due to the decay products

of ^{222}Rn gas generated by uranium decay in soil [2]. Rain can accordingly increase the level of background radiation by releasing trapped radon gas and other radioactive sources in the soil.

To address geologic background changes in our simulated datasets, gamma-ray background spectra were simulated with geologic material compositions from Albuquerque, New Mexico; Atlanta, Georgia; Austin, Texas; Chicago, Illinois; Knoxville, Tennessee; and Miami, Florida [52]. These locations were chosen because of their geologic differences. Temporal changes in background were not addressed in this work.

2.1.4 Effect of Temperature on Calibration and Resolution

The calibration of a NaI(Tl) detector can vary due to drifts in component voltages and drifts in the crystal's and electronic component's temperature. Calibration drift has been identified as a key factor in the poor performance of RIIDs [13]. The distribution of the intensities of different NaI(Tl) scintillation light decay processes [53] and decay time constants of these processes [7] are sensitive to temperature. These factors lead to appreciable changes in relative photopeak position with temperature, shown in Figure 2.11, and detector resolution, shown in Figure 2.12.

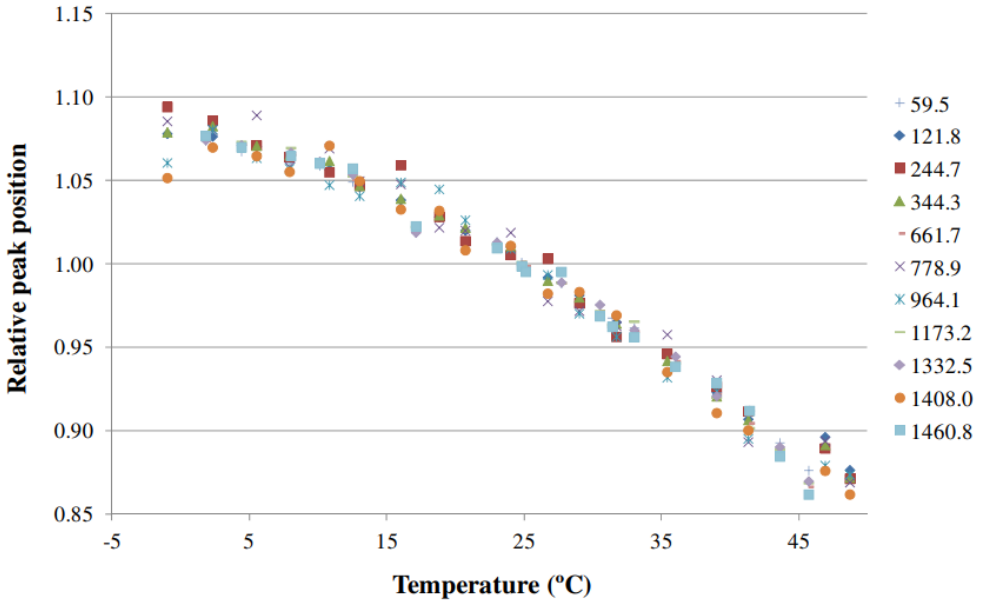


Figure 2.11: Temperature vs relative photopeak position for an ORTEC Model 905-3 2x2 NaI(Tl) detector. Reproduced from [6].

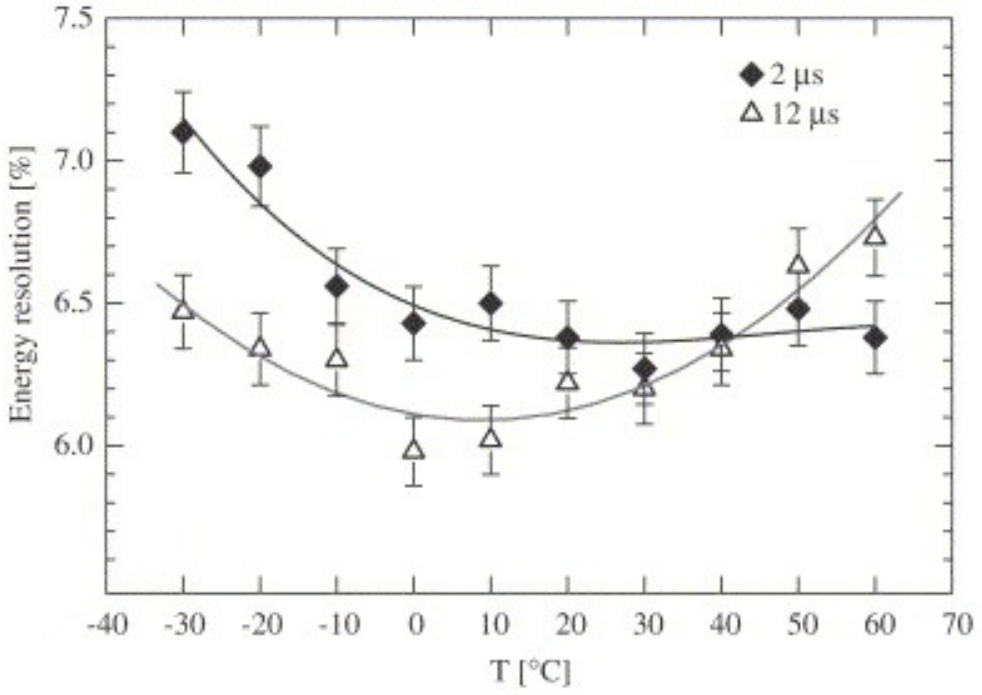


Figure 2.12: The energy resolution of 662 keV γ-rays measured with a 25mm x 30mm NaI(Tl) crystal using 2 μs and 12 μs peaking times. Reproduced from [7].

While automated gain stabilization methods exist, many require a clear photopeak in the spectrum to calibrate from. This can be achieved with measurement times long enough to identify a background photopeak, attaching an external radioactive source to the detector, or by adding a reference light source. Measurement times long enough to identify a background photopeak may be infeasible when checking for radioactive material in cargo containers or vehicles at boarder crossings. Calibration steps requiring active participation by the user, such as calibrating with an external radioactive source, may be ignored altogether. Adding an internal source of radiation adds additional noise to the spectrum. For these reasons, we added ranges of detector calibration gains to our simulated spectra. The smallest gain range considered, [0.9, 1.1], addresses gain changes based on expected operational temperatures found in Figure 2.11. The larger gain range considered, [0.8, 1.2], addresses temperature changes as well as larger calibration deviations from operator error or negligence.

2.2 Machine Learning

Machine learning models can be thought of as black box functions that transform some input into some output given a provided training dataset. An ANN's capacity describes how complicated this function can be. High capacity models can learn complicated relationships between inputs and outputs, but are sensitive to overtraining. Overtraining occurs when the model memorizes a training dataset and loses the ability to map inputs from outside the training dataset to correct outputs. Low capacity models fail to learn the relationship between a given input and output dataset. A model's capacity is related to the number of its free parameters. Rigorously determining a model's capacity is an active area of machine learning research.

Machine learning tasks often require different architectural and training hyperparameters for optimal performance on a tasks of different complexity [54]. To investigate how hyperparameters differ for gamma-ray spectroscopy, in Chapter 3 we optimize architectures for isotope classification tasks of increasing difficulty. Analyzing optimal hyperparamters also allows us to determine model capacity requirements for increasingly difficult problems.

In this section, we describe the general theory behind how neural networks operate, how to train them, and how to optimize their architectures for a task. We also give an introduction to neural network architectures studied in this work and describe parameters that affect their training.

2.2.1 General Neural Network Architecture

An artificial neural network (ANN) is a mathematical model that attempts to map an arbitrary function from $\mathbb{R}^M$ to $\mathbb{R}^N$, where M and N are positive integers. An ANN accomplishes this by mimicking biological neurons. One example of an ANN architecture is shown in Figure 2.13. This ANN has N neurons in input layer A, J neurons in layer B, and K neurons in output layer C. Layer B is called a hidden layer because it is not directly observed in the input or output layers. Neurons in adjacent layers are connected by a weight matrix, represented in Figure 2.13 by arrows connecting neurons.

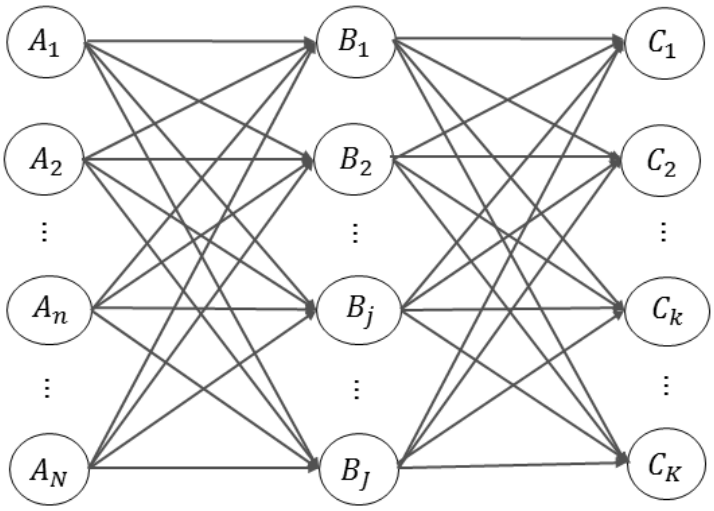


Figure 2.13: Example ANN with input neurons A_n, hidden neurons B_j, and output neurons C_k [8].

Similar to a biological neuron, the ANN neuron receives input stimuli, performs an operation using it, and returns a signal. The structure and equation governing the operation of an individual neuron is shown in Figure 2.14.

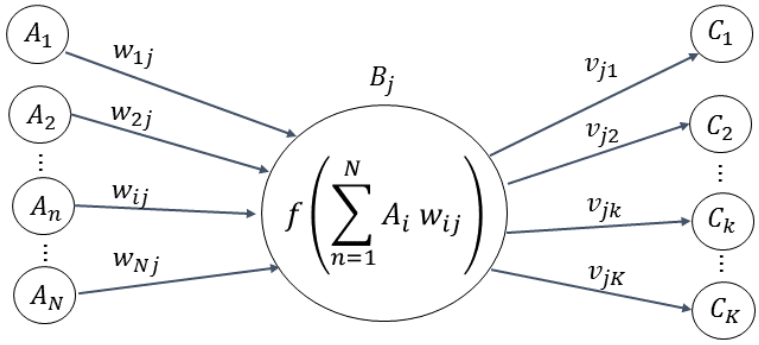


Figure 2.14: Summary of the operation of a single neuron [8].

As seen in Figure 2.14, each neuron operates by summing the products of the previous layer's values (A_1, A_2, ... A_N) and each individual weight (w_{1j}, w_{2j}, ... w_{nj}) connecting the neurons. This summation is analogous to the stimulus a biological neuron receives from its dendrites. A a non-linear activation function, f, operates on the total stimulus. The output signal, also known as the neurons activation, is then passed to the next layer of

the ANN where the process repeats. An ANN may be trained by finding a function, f, that solves

$$\underset{\mathbf{W}}{\operatorname{argmin}} \ \mathrm{E}(f(\mathbf{X}; \mathbf{W}), \mathbf{T}) \tag{2.3}$$

where

$$\begin{aligned} \mathrm{E} &= \text{error metric} \\ \mathbf{W} &= \text{weight matrix} \\ \mathbf{T} &= \text{target values} \\ \mathbf{X} &= \text{input data.} \end{aligned}$$

Except in simple cases, Equation 2.3 cannot be solved analytically. Numerical methods for solving this equation include gradient descent through the back-propagation of errors [25], genetic learning algorithms [55], and Newton's method [56]. In this work, we use a gradient descent implementation called Adam (adaptive moment estimation) [57]. Adam is described in section 2.2.4.3.

2.2.2 Neural Network Training

The most common ANN training method is applying gradient descent to the backpropagation of errors. The gradient we attempt to minimize is the derivative of an error metric with respect to the network's weights and biases. These gradients show how the networks weights and biases need to change,

$$\Delta w_j = -\eta \frac{dE(D)}{dw_j} \tag{2.4}$$

$$\Delta b_j = -\eta \frac{dE(D)}{db_j} \tag{2.5}$$

where

$$
\begin{aligned}
w_j &= j^{th} \text{ weight} \\
b_j &= j^{th} \text{ bias} \\
\mathrm{E} &= \text{error metric} \\
\eta &= \text{learning rate,}
\end{aligned}
$$

to reduce the error metric for some training data,

$$D = (\boldsymbol{x}_1, t_1), ..., (\boldsymbol{x}_n, t_n), ..., (\boldsymbol{x}_N, t_N), \tag{2.6}$$

where

$$
\begin{aligned}
\boldsymbol{x}_n &= n^{th} \text{ training example} \\
t_n &= n^{th} \text{ training target.}
\end{aligned}
$$

The learning rate and its affect on training are discussed further in section 2.2.4.1. Using these equations, the gradients can be repeatedly measured to iteratively update the weights. Updating the network with the entire dataset only once is referred to as one training *epoch*. Due to computational constraints involving memory, it is often infeasible to calculate the update rules in Equations 2.4 and 2.5 for the entire training dataset. A common technique used to train on large datasets is to iteratively update weights using disjoint subsets of the training data, called *minibatches*.

Training occurs until an early stopping condition is met. Early stopping works by first removing a portion of the training data and defining it as validation data. The ANN is trained using the new training data while we monitor an error metric against the training and validation datasets. Early stopping conditions include ending training when a threshold on an error metric measured in the validation dataset is reached or when the error metric has not improved (either by some factor or absolutely) in a fixed number of epochs, called the early stopping patience.

Early stopping is needed to end training when the model begins to overfit or learning plateaus. Overfitting occurs when the validation dataset's error increases and the training dataset's error decreases, illustrated in Figure 2.15.

This indicates that the model is memorizing the training data and not *generalizing* to examples not seen during training. Generalization is a term used to describe a model's performance data outside the training dataset. Learning can also plateau when the ANN's weight updates are not large enough to escape a minimum in the error metric measured on the training dataset.

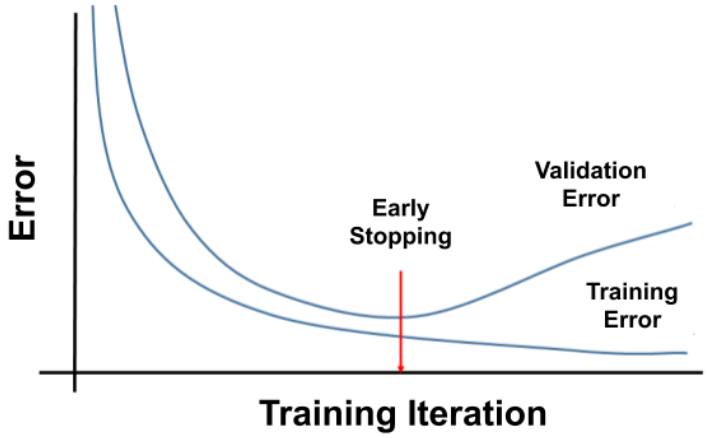


Figure 2.15: Example training and validation error curves.

A common technique for choosing training and validation data is k-folds cross validation. Figure 2.16 illustrates 5-folds cross validation. In this process, $\frac{1}{k}$ of all available data is split into a validation dataset and the rest is defined as the training dataset. The model is then trained using the training data and the validation dataset's error is recorded. The process of splitting the dataset is repeated k times. The average error metric of each trained model on the validation dataset is used to estimate the generalization capabilities of the overall model. Typical values for k are 5 or 10. To reduce computational requirements, we use 5-folds cross validation in our hyperparameter searches and final model training.

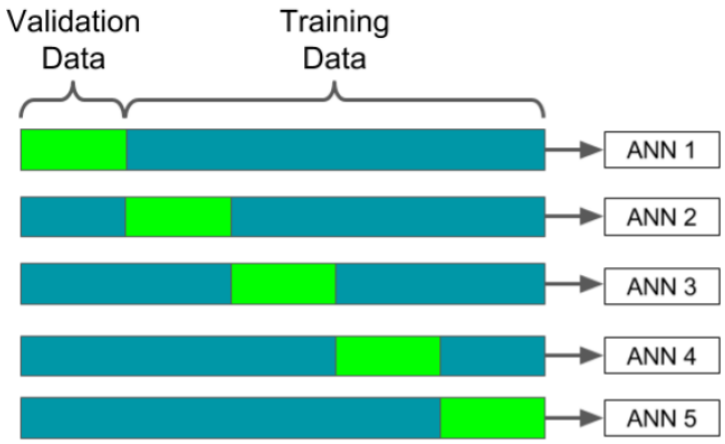


Figure 2.16: An example of k-folds cross validation.

2.2.3 Simple Neural Network Example

An example of a simple one-layer ANN is shown in Figure 2.17. This ANN is a function that takes two real-valued inputs (x_1 and x_2) and performs the operation shown in Figure 2.14. The weights in the hidden layer connecting the i^{th} input neuron to the j^{th} output neuron are represented by w_{ij}. ANNs where nodes in each subsequent layer are connected to each node in the previous layer are referred to as densely connected. An extra node, called the bias, is added to make the layer affine. Without the bias, the layer would be linear, meaning an input vector of zeros could only be mapped to another vector of zeros. The bias allows for more expressive mappings, adding to the representational power of the model. The bias activation is set to a constant value of 1, allowing the bias to be trained by changing the weights connecting the bias to the next layer. Using the hyperbolic tangent function, the network outputs for each class y_1, y_2, and y_3 range [-1, 1]. Using this function, if an input results in the y_j^{th} output neuron rising above zero, that input can be classified as part of the j^{th} class (or as not a part of the j^{th} class if the output is below zero). Other functions that map outputs values to arbitrary bounds (e.g. other sigmoid functions or an unbounded linear function) can be used. The equation for the output corresponding to the j^{th} class is

$$y_j = \tanh(\sum_i x_i w_{ij} + b_j) \tag{2.7}$$

where

$y_j = j^{th}$ output activation

$x_i = i^{th}$ input from the previous layer

b_j = weight connecting the j^{th} output to the bias neuron.

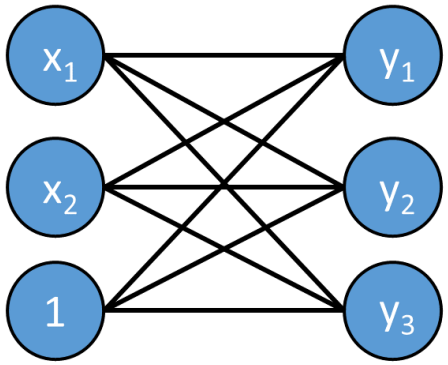


Figure 2.17: Example of a single-layer neural network with two inputs (x_1 and x_2), three classes (y_1, y_2, y_3), and a bias neuron set to one.

The geometry of the above network's operation is illustrated for an example dataset in Figure 2.18. This dataset is composed of three classes: red, green, and blue. The axes that define this dataset are the inputs to the single layer network in Figure 2.17, (x_1,x_2). If $\mathbf{W}$ is defined to be a vector with elements (w_{11}, w_{21}), a line can be defined perpendicular to $\mathbf{W}$ and shifted by $\frac{b_1}{||\mathbf{W}||}$ from the origin in the direction of $\mathbf{W}$. Given appropriate values for w_{11}, w_{21}, and b_1, a line that separates the blue class from the non-blue class can be created. Any point on the -$\mathbf{W}$ side of the line will have $y_1 < 0$, allowing for classification. Similarly, a separating line for the red class using w_{12}, w_{22}, and b_2 and a separating line for the green class using w_{13}, w_{23}, and b_3 can be constructed.

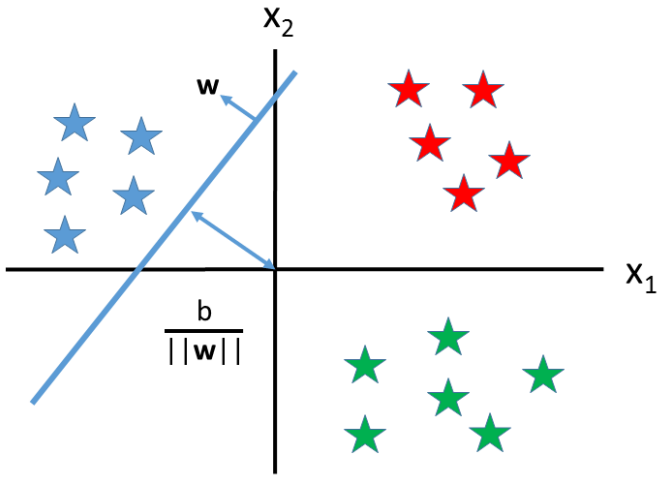


Figure 2.18: A dataset describing a three class function. Each class is represented by a different color.

The classes in this example dataset are linearly separable, meaning lines can be drawn completely separating each class. If the classes were not linearly separable, additional hidden layers would be necessary to compute the function. It has been shown that additional hidden layers allow the creation of arbitrary decision boundaries [58].

Backpropagation uses the chain rule to find these derivatives for some training data and target. Applying backpropogation in a single-layer ANN requires finding an expression for the derivative of an error function with respect to the networks weights [59]. For example, to apply backpropogation when using the mean squared error (MSE) function,

$$E_{MSE} = \frac{1}{N}\sum_{n=1}^{N}(t_n - y_n)^2, \tag{2.8}$$

on the network described in Equation 2.7, we apply the chain rule to find an expression for

$$\frac{dE_{MSE}}{dw_j} = \frac{dE_{MSE}}{dy_i}\frac{dy_i}{dw_j}. \tag{2.9}$$

Equation 2.9 can be solved by first evaluating $\frac{dE_{MSE}}{dy_i}$,

$$\begin{aligned}\frac{dE_{MSE}}{dy_i} &= \frac{d}{dy_i}\frac{1}{N}\sum_{i=1}^{N}(t_i - y_i)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\frac{d}{dy_i}(t_i - y_i)^2 \\ &= -\frac{2}{N}\sum_{i=1}^{N}(t_i - y_i).\end{aligned} \tag{2.10}$$

Evaluating $\frac{dy_i}{dw_j}$,

$$\begin{aligned}\frac{dy_i}{dw_j} &= \frac{d}{dw_j}\tanh(\boldsymbol{w}'\boldsymbol{x}_i + b) \\ &= \tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b)\frac{d}{dw_j}(\boldsymbol{w}'\boldsymbol{x}_i + b) \\ &= \tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b)x_{ij}.\end{aligned} \tag{2.11}$$

where x_{ij} is the j^{th} index of the i^{th} training vector. The update rule for the bias is found using

$$\frac{dE_{MSE}}{db_j} = \frac{dE_{MSE}}{dy_i}\frac{dy_i}{db_j}. \tag{2.12}$$

The expression for $\frac{dE_{MSE}}{dy_i}$ is known from Equation 2.11. Evaluating the other derivative in 2.12 we find

$$\begin{aligned}\frac{dy_i}{db_j} &= \frac{d}{db_j}\tanh(\boldsymbol{w}'\boldsymbol{x}_i + b) \\ &= \tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b).\end{aligned} \tag{2.13}$$

Finally, the gradient of the error metric can be computed for a single weight,

$$\frac{dE_{MSE}}{dw_j} = -\frac{2}{N}\sum_{i=1}^{N}(t_i - y_i)\tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b)x_{ij}, \tag{2.14}$$

and a single bias,

$$\frac{dE_{MSE}}{db_j} = -\frac{2}{N}\sum_{i=1}^{N}(t_i - y_i)\tanh'(\boldsymbol{w}'\boldsymbol{x}_i + b). \tag{2.15}$$

Gradient descent can be applied to an ANN with multiple layers. The input to the l^{th} layer given some training example $\boldsymbol{x}$ in a network with L layers is

$$\boldsymbol{z}^{x,l} = \boldsymbol{w}^l \boldsymbol{a}^{x,l-1} + \boldsymbol{b}^l \tag{2.16}$$

where the activation from the $(l-1)^{th}$ layer is

$$\boldsymbol{a}^{x,l-1} = f^{l-1}(\boldsymbol{z}^{x,l-1}) \tag{2.17}$$

and where $\boldsymbol{w}^l$ = weight matrix of layer l,

$\boldsymbol{b}^l$ = bias vector of layer l,

f^{l-1} = non-linear activation function used in layer (l-1).

Defining the output error as

$$\delta^{x,L} = \frac{\partial C}{\partial a^{x,L}} \odot \frac{\partial f^{l-1}(\boldsymbol{z}^{x,l-1})}{\partial \boldsymbol{z}^{x,l-1}}, \tag{2.18}$$

where $\odot$ is the element-wise product, the output error can backpropagate to previous layers. For each $l = L-1, L-2, ..., 2$ an error can be defined by

$$\delta^{x,l} = ((\boldsymbol{w}^{l+1})^T \delta^{l+1}) \odot \frac{\partial f^l(\boldsymbol{z}^{x,l})}{\partial \boldsymbol{z}^{x,l}}. \tag{2.19}$$

The gradient of the cost function as a function of each individual weight and bias can now be defined as

$$\frac{\partial E}{\partial w_{jk}} = a_k^{l-1} \delta_j^l \tag{2.20}$$

and

$$\frac{\partial E}{\partial b_j} = \delta_j^l. \tag{2.21}$$

Using Equations 2.20 and 2.21, the weights can be updated for a single backpropogation iteration.

2.2.4 Hyperparameters

In addition to the weights connecting neurons, *hyperparameters* can modify ANN behavior. Hyperparameters determine both the network's structure (number of layers, number of nodes in each layer, activation function for each layer) and how the model learns (learning rate, momentum, loss function). The following section discusses various hyperparameters and their effects on ANN learning.

Methods for hyperparameter optimization are discussed in Section 2.2.5. The hyperparameter ranges explored in this dissertation are base on ranges used in our published work [8, 18, 17].

2.2.4.1 Learning Rate

The learning rate , η, affects the magnitude of each weight update. If η is too small, the network will learn slowly and training will be inefficient. If η is too large, the network will fail to learn, either by converging to a non-extremum or by diverging. An example of a small and large learning rate are shown in Figure 2.19

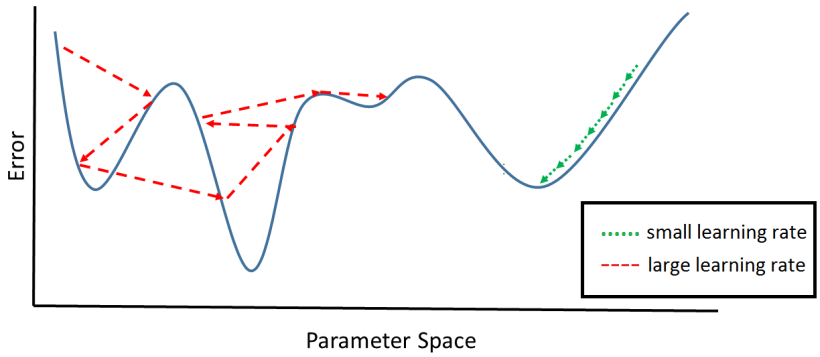


Figure 2.19: Example training paths for a large learning rate and a small learning rate.

There are many methods to modify η to encourage more efficient learning. One method to increase the speed of learning is to start with a large η and decrease η as a function of iteration number. Ideally, this method would lead to quick initial learning when far from an optimum and slower learning near an optimum to more finely explore it. The difficulty with this method is the

requirement for a function that slows the learning rate efficiently for a given problem.

2.2.4.2 Learning Momentum

Another method to speed up learning is to add a momentum hyperparameter, μ, to the weight update algorithm [60],

$$\Delta w_{ij}(n) = -\eta \frac{dE_{MSE}}{dw_j} + \mu \Delta w_{ij}(n-1). \tag{2.22}$$

Similar to the goal of slowing learning over time described above, the momentum term attempts to slow learning near optima. The momentum will be large when the weights are updated at large steps, far from an optima, but will decrease near an optimum, allowing slower learning near an optimum. The learning momentum term is employed by the Adam optimizer, which is described in the following section.

2.2.4.3 Training Algorithms

There are many ANN training algorithms that employ various learning rate schedules and momentum functions. These algorithms include but are not limited to: Nesterov's accelerated gradient [61], simulated annealing [62], ADADELTA (An Adaptive Learning Rate Method) [63], and Adam [57].

In this work, we chose the Adam optimizer as implemented by TensorFlow [64] as the training algorithm. The Adam optimizer is widely used due to training speedups and performance improvements on popular benchmark datasets like MNIST (Modified National Institute of Standards and Technology) [65], IMDB (Internet Movie Database) movie reviews [66], and CIFAR-10 (Canadian Institute For Advanced Research) [67]. Another benefit of Adam is the introduction of effectively only one hyperparameter, the learning rate.

The Adam optimizer update rule is described below. For the following, g_t is the gradient of the error function with respect to the network parameters at iteration t,

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{2.23}$$

is the estimate of the mean of the gradient at iteration t and

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.24}$$

is the estimate of the variance of the gradient at iteration t. For the following, the variables β_1 and β_2 are parameters called decay rates, ϵ is included for numerical stability, and θ_t represents the network parameters at iteration t. As described by Kingma and Lei Ba, the default values for β_1, β_2, and ϵ are 0.9, 0.999, and 10^{-8} respectively [57]. These values were seen to work well for a variety of problems. While these hyperparameters can also be tuned, it has been shown that the default values work well for a variety of network architectures and datasets [57]. The bias-corrected first moment estimate is given by

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.25}$$

and the bias-corrected second moment estimate is given by

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{2.26}$$

Finally, the weight update equation is computed as

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t. \tag{2.27}$$

2.2.4.4 Cost Functions

The choice of cost function used in an ANN depends on the task the network is attempting to learn. Tasks that require the network's output be real numbers typically use the MSE function and multi-class classification tasks use the mean cross entropy (MCE),

$$E_{MCE} = -\frac{1}{N} \sum_{n=1}^{N} y_n \log(\hat{y}_n). \tag{2.28}$$

The MSE function is appropriate when targets are any real number, as in a regression problem. The MSE cost function is used in the uranium enrichment regression problem in Chapter 5. This cost function is also used when training the autoencoders in this work. Autoencoders, described in

more detail in Section 2.2.6.2, are ANNs that attempt to reconstruct an input signal.

Another cost function is cross entropy. This function measures the similarity between two probability distributions. By using cross entropy with the softmax output function,

$$softmax(z_j) = \frac{\exp(z_j)}{\sum_{k=1}^{K} \exp(z_k)}, \tag{2.29}$$

we can calculate the posterior probability of each class in a classification problem [68]. The cross entropy cost function and softmax output function are used in the source interdiction classification problem analyzed in Chapters 3 and 4.

2.2.4.5 Weight Regularization

Weight regularization is a hyperparameter that penalizes the ANN based on the magnitude of its weights. Weight magnitudes are related to the complexity of the ANN. Adding weight regularization attempts to limit complexity and the probability of overfitting.

A common method of incorporating weight regularization is by adding an L_n regularization term to the error function,

$$\tilde{E} = E + \sum_i \lambda w_i^n. \tag{2.30}$$

Common values for n are 1 and 2. Adding weight regularization allows the magnitude of the weights to increase only when there is a comparable reduction in the unmodified error function.

In Equation 2.30, w_i is the weight between each neuron in the ANN and λ is the regularization strength hyperparameter. A larger λ will force the ANN to prefer smaller weights connecting the neurons. If the parameter λ is too small, the unbounded model complexity may fit only the training data. If the parameter λ is too large, the ANN will only minimize the L_n error, failing to learn.

In this dissertation, L_2 weight regularization is added to the densely connected portions of the non-autoencoder networks.

2.2.4.6 Neuron Dropout

Another method to reduce model complexity is by adding a *neuron dropout rate* hyperparameter. Neuron dropout is the process of temporarily removing a random set of neuron from the ANN architecture during each training iteration [69].

Almost always, taking the average output of more than one separately trained ANN improves the performance of the ANN [69]. By applying dropout at each neuron with the same probability throughout training, the ANN's architecture changes every iteration. This makes neuron dropout a cost efficient way to effectively average many different ANN architectures, improving performance.

In this dissertation, neuron dropout is added to the densely connected portions of the non-autoencoder networks.

2.2.4.7 Data Augmentation

Machine learning algorithms perform better with more data. To increase the amount of data available, before each training iteration the input data can be randomly modified using physically realistic transformations. This process is called online *data augmentation*. Data augmentation can be performed either online or offline. Online data augmentation refers to applying the augmentation during each training batch, ensuring the network never sees the same data twice. If augmenting the data is computationally expensive, this method can slow training - potentially prohibitively. Offline data augmentation is used to expand the dataset before training. An example data augmentation applied to an image of a cat is shown in Figure 2.20.

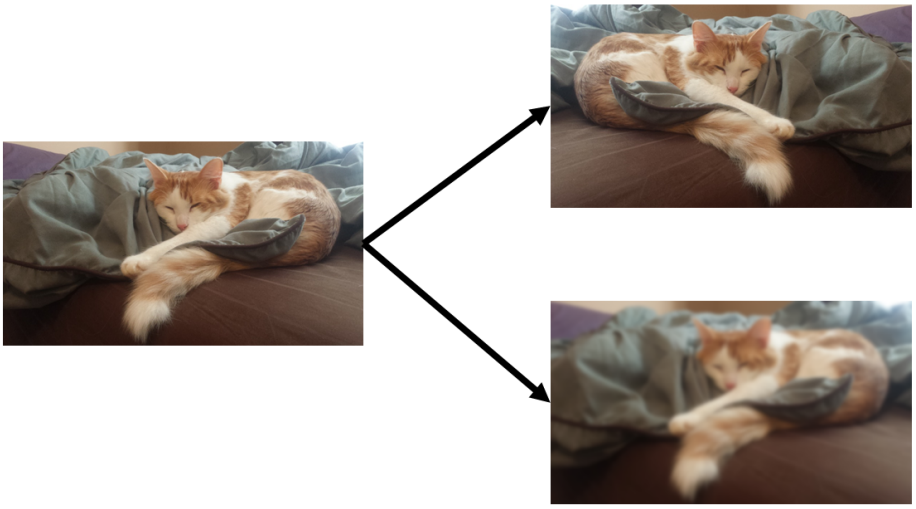


Figure 2.20: Two examples of data augmentation using an image of a cat. The image to the left is the original. The top right image is augmented using a horizontal flip. The bottom right image is augmented using blur.

In this work we employ offline data augmentation to expand a dataset of gamma-ray spectra obtained from a photon transport software. We augment data by changing the detectors calibration, ratio of source to background counts, background count rate, and measurement time.

2.2.5 Hyperparameter Optimization

ANN hyperparameters require optimization. Optimizing these hyperparameters leads to more efficient training and better generalization performance. Methods to perform ANN hyperparameter optimization include manual optimization, exhaustive grid search, and random parameter search.

Manually optimizing parameters is necessary when developing a novel algorithm. This involves changing hyperparameters and observing training performance and validation dataset error. Ideally, the ANN should train quickly with low error on a validation dataset. For many parameters, 'rule of thumb' values exist that can be used to find parameters that work to some degree. Due to the large hyperparameter space used in this work, a manual search is cost prohibitive.

Once a range of parameters is determined through a manual search, multiple methods are available to explore the parameter space for an optimal solution. One method is an exhaustive grid search. In a grid search, the

parameter space is divided into a uniform grid and the joint performance of all parameters is tested. The grid search method is generally ineffective for two reasons. First, neural networks may have a large number of hyperparameters that need to be explored, and the computational requirement to explore the hyperparameter space increases exponentially with increasing hyperparamters. Second, in practice only a few hyperparameters dominate performance, but the dominating hyperparameters are different for different applications. A grid search may under-represent the importance of key hyperparamters, as seen in Figure 2.21. While this method works, it has been shown that a random search in the hyperparameter target domain finds better hyperparameters quicker than testing equally distributed points in the chosen range [54]. It can also be shown that given 60 random samples over some space with a finite minimum, the minimum of those 60 random samples is within 5% of the true minimum with 95% probability [70]. In this work, we employ random hyperparameter searches using 256 randomly chosen hyperparameters to ensure this condition is met.

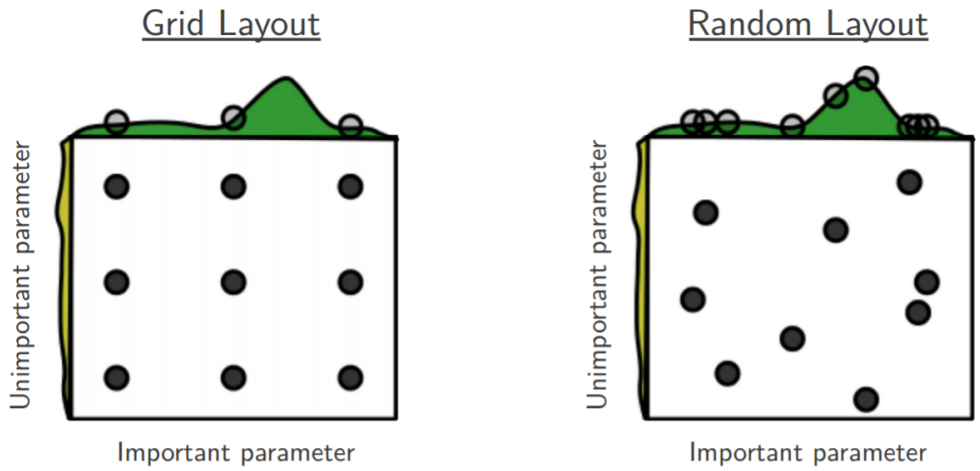


Figure 2.21: A comparison between a grid search and a random search for hyperparameter optimization. In this example, performance is strongly tied to one hyperparameter. The green function represents the effect of an important hyperparameter on a cost function while the yellow function represents the effect for an unimportant hyperparameter. Figure reproduced from [42].

2.2.6 Additional Machine Learning Models

In addition to dense models, we also include convolutional and autoencoder models in this work. Convolutional models are used in this work because they assume the input data have local structure. Local structure refers to the fact that values around a given point in some data are related. In gamma-ray spectra local structure manifests itself in the shape of peaks and Compton continua.

Autoencoders, machine learning models that attempt to reconstruct their input, are included to investigate their use for model pretraining. By training autoencoders to reconstruct gamma-ray spectra, we pretrain their weights for gamma-ray spectroscopy. This pretraining may allow the networks to learn a task more effectively.

2.2.6.1 Convolutional Neural Networks

Before the popularization of machine learning, many pattern recognition algorithms relied on hand-crafted feature extractors (e.g. manually creating features using edge and line detectors for face recognition) and a simple trainable classifier like a linear model or naïve Bayes. For example, using 1D convolutions to extract features from gamma-ray spectra, also known as the wavelet method, has been previously studied for isotope identification [36, 71, 72, 73]. There are a number of issues with creating and using custom feature extractors. Robust feature extraction algorithms often require extensive domain knowledge - making them expensive to create. There is rarely a guarantee that these features will be optimal for classification or regression tasks. A classifier created around suboptimal features has the potential to perform poorly in real-world conditions or similar problems.

Deep learning algorithms use a dataset to learn optimum feature extraction and classification techniques simultaneously. CNNs do this by using locally connected convolutional filters instead of fully connected weights. DNNs are fully connected architectures, which means each activation from the previous layer is considered in the next. This leads to redundancy in problems where the input has local structure, like in image recognition or signal processing. Additional CNN layers add hierarchies of representations, meaning that early layers use simple features and deeper layers combine them into more

complex features. The magnitude of abstraction can be controlled by tuning the number of convolutional layers in a model. Because low-level features are usually shared among the classes in some data (e.g. edge detection in hand-written digits) the convolutional filters can be shared among classes. This weight sharing decreases the number of parameters in the model which decreases the probability of overfitting. Weight sharing also makes the model more robust against transformations in the input (e.g. rotations, scaling, and translations).

Figure 2.22 shows one of the first successful CNN architectures, LeNet-5 [9], which classified 32x32 images of handwritten digits using two convolution and average pooling layers followed by two dense layers and an output. This was the first example of a machine learning algorithm creating custom filters and a classification model for the MNIST dataset.

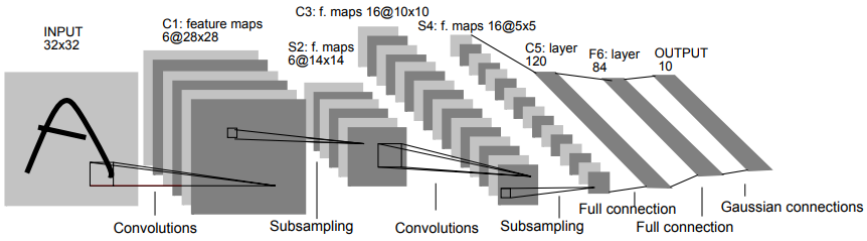


Figure 2.22: Architecture of LeNet-5, a CNN created for digit recognition. Image reproduced from [9]

2.2.6.2 Autoencoders

Autoencoders were introduced by Hinton [74] as a generalized nonlinear dimension reduction technique. An autoencoder is a neural network designed to learn a compact representation of some input. This is accomplished by simultaneously training an *encoding network* and a *decoding network*. An example of this can be seen in Figure 2.23. The encoding ANN reduces an $n-$dimension input signal, X, to a $m-$dimension signal, z, where $m < n$. The decoding ANN takes the encoded signal, z, and reproduces the input signal, X'. Once trained, the encoder can pre-train other neural networks or serve as a feature extractor [75, 76].

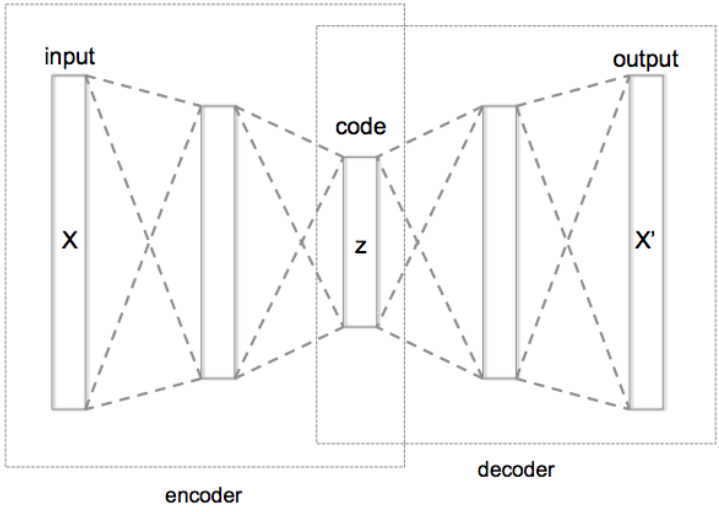


Figure 2.23: An example of an autoencoder. Image reproduced from [10].

Methods to encourage the network to learn useful features include forcing the autoencoder to perform denoising [77, 78] and by using an encoding smaller than the input signal (also called an undercomplete autoencoder). Denoising autoencoders corrupt their input with noise, making the autoencoder learn more robust features than those used for simply copying the input.

Other commonly used feature extraction methods are principal component analysis (PCA) [38], and linear discriminant analysis (LDA) [79]. PCA attempts to reduce the dimensionality of a dataset into linearly uncorrelated variables. Using a few of these principle components, the data may be represented in a reduced space that contains most of the information present in the original data. Another linear feature extraction method is LDA. LDA is a supervised feature extraction method that finds linear combinations of features that can be used to separate classes. An example of a non-linear feature extraction methods is kernel PCA. Kernel PCA applies a non-linear transform to the input space and applies PCA to the data in this transformed space. For kernel PCA to perform well, the correct kernel must be chosen for a given problem, which is a non-trivial task.

2.3 annsa

As a part of this work we developed the open source (under the BSD 3 license) software package `annsa` [51]. `annsa`'s main modules are shown in Figure 2.24. The dataset generation module leverages gamma-ray transport software applications to quickly prototype and generate training datasets. The machine learning model construction module uses TensorFlow [64] and Keras [80] to build machine learning models for spectroscopic tasks. `annsa` contains machine learning models designed for tasks such as: classifying spectra based on an isotope library, quantifying an object's uranium enrichment, and extracting features from a spectrum using autoencoders. Machine learning models included in `annsa` are dense neural networks (DNNs), convolutional neural networks (CNNs), dense autoencoders (DAEs), and convolutional autoencoders (CAEs).

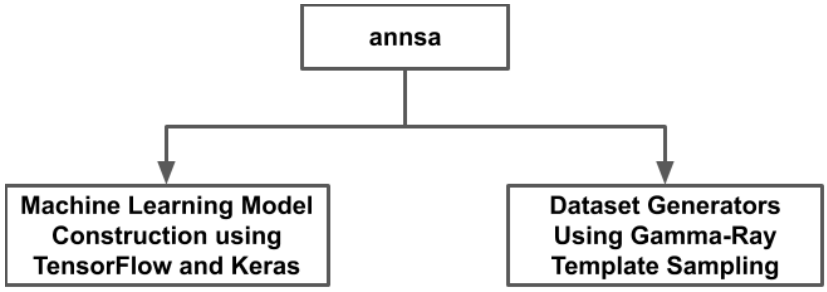


Figure 2.24: The main modules in the `annsa` package.

Additional machine learning models and spectroscopic dataset construction methods can be added to `annsa`. To make the integration of new and updating of existing components easier, we also included unit tests using `pytest` [81] and the continuous integration framework Circle CI. Circle CI runs `annsa`'s unit tests when new code is uploaded to it's GitHub repository. This process ensures new functionality will not affect `annsa`'s current functionality.

The open source software stack `annsa` uses include: TensorFlow [64] and Keras [80] for the machine learning model creation and training; NumPy [82] and pandas [83] to manipulate and store data; and scikit-learn [84] for prepossessing and miscellaneous training tasks. `annsa` can be installed in a UNIX environment using the following commands:

```
git clone https://github.com/arfc/annsa.git
```

```
cd annsa
python setup.py install
```

2.3.1 Dataset Generation

The dataset generation process begins by creating a dataset of noiseless template spectra. A gamma-ray transport software package and a software package that simulates the response from a gamma-ray detector are used to generate the templates. The gamma-ray transport software package simulates how photons from each isotope scatter in an environment to reach the detector. Transport software package that could be used for this include GADRAS-DRF [52], MCNP [85], or GEANT4 [86]. The detector response software accounts for the calibration and Gaussian energy broadening of the detector.

`annsa` uses separate source and background template spectra datasets (csv formatted). These templates are simulated without Poisson noise. Poisson noise can be removed from a spectrum by simulating spectra with sufficient counts to minimize channel-to-channel variance. Columns in each template dataset correspond to simulation settings. Example rows from the source and background template csv files used in this work are shown in Figure 2.25. Source height, distance, and shielding are assumed to have no effect on the background templates and omitted from the background template dataset. Instead, the background template dataset includes entries for the location simulated, resolution of the detector, and the inclusion of cosmic radiation background. Cosmic radiation is assumed to be negligible and is not simulated for the background templates in this work.

Source

isotope	sourceheight	sourcedist	shielding	shieldingdensity	fwhm	channel0	...	channel1193
60CO	100.0	175.0	alum	1.82	7.0	212.789		0.0

Background

location	fwhm	cosmic	channel0	...	channel1193
albuquerque	7.5	0	1241.135		0.0

Figure 2.25: Example csv entries for the source and background template datasets.

A wide variety of realistic gamma-ray spectra can be generated using noiseless templates without the need to excessively run computationally expensive transport software applications. Spectra can be simulated by combining source and background templates in desired amounts,

$$C_{total}(Ch) = tb_{cps}B(Ch) + ts_{cps}\sum_{n=1}^{N} c_n C_n(Ch) \tag{2.31}$$

where

$$\begin{aligned} C_{total}(Ch) &= \text{Counts in channel Ch for the simulated spectrum} \\ B(Ch) &= \text{Counts in channel Ch for the background template} \\ C_n(Ch) &= \text{Counts in channel Ch for n}^{th} \text{ source template} \\ c_n &= \text{Relative contribution from the n}^{th} \text{ source template} \\ b_{cps} &= \text{Background count rate measured by the detector [cps]} \\ s_{cps} &= \text{Total source count rate measured by the detector [cps]} \\ t &= \text{Measurment time [s].} \end{aligned}$$

Source template probability density functions are created by rebinning a chosen template, removing channels above the 1024^{th}, setting counts defined by the low level discriminator to zero, and finally normalizing the spectrum by it's total counts. This process is illustrated in Figure 2.26 for a background and ^{60}Co template spectrum. `annsa` implements a function that chooses templates based on a users input.

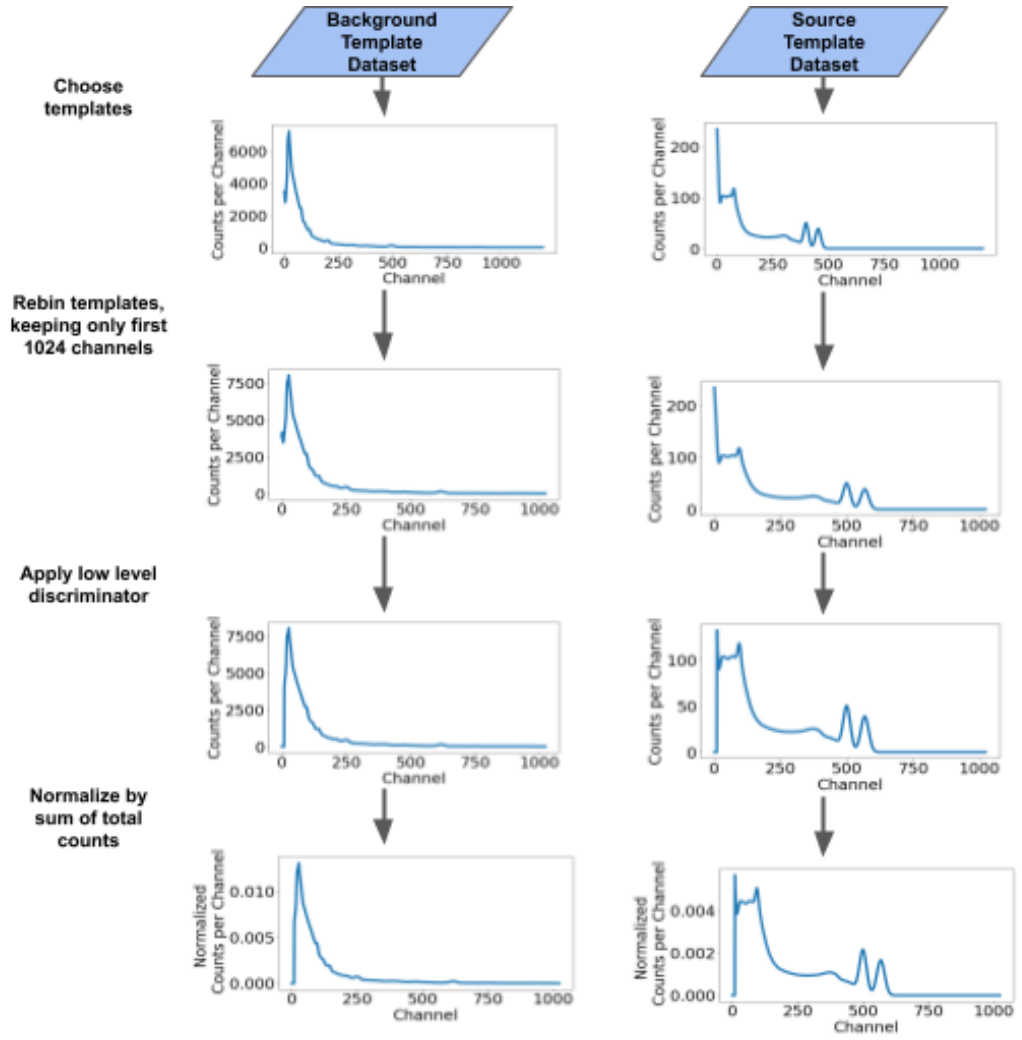


Figure 2.26: An example of the procedure used to normalize a template spectrum.

After the normalization process, spectra are combined based on total counts desired in the source and background signals. An example of combining background and ^{60}Co probability density function templates is shown in Figure 2.27. In this illustration, the signal to background ratio is 5 and the expected total number of counts in the final spectrum is 6000. Normalized templates are scaled by the appropriate amounts, combined, and sampled using a Poisson distribution to yield spectra with realistic counting statistics. These spectra and machine learning target values (which isotope is in a spectrum, relative quantities of each isotope in a spectral mixture, ^{235}U % enrichment) are stored and saved in a NumPy array for future machine

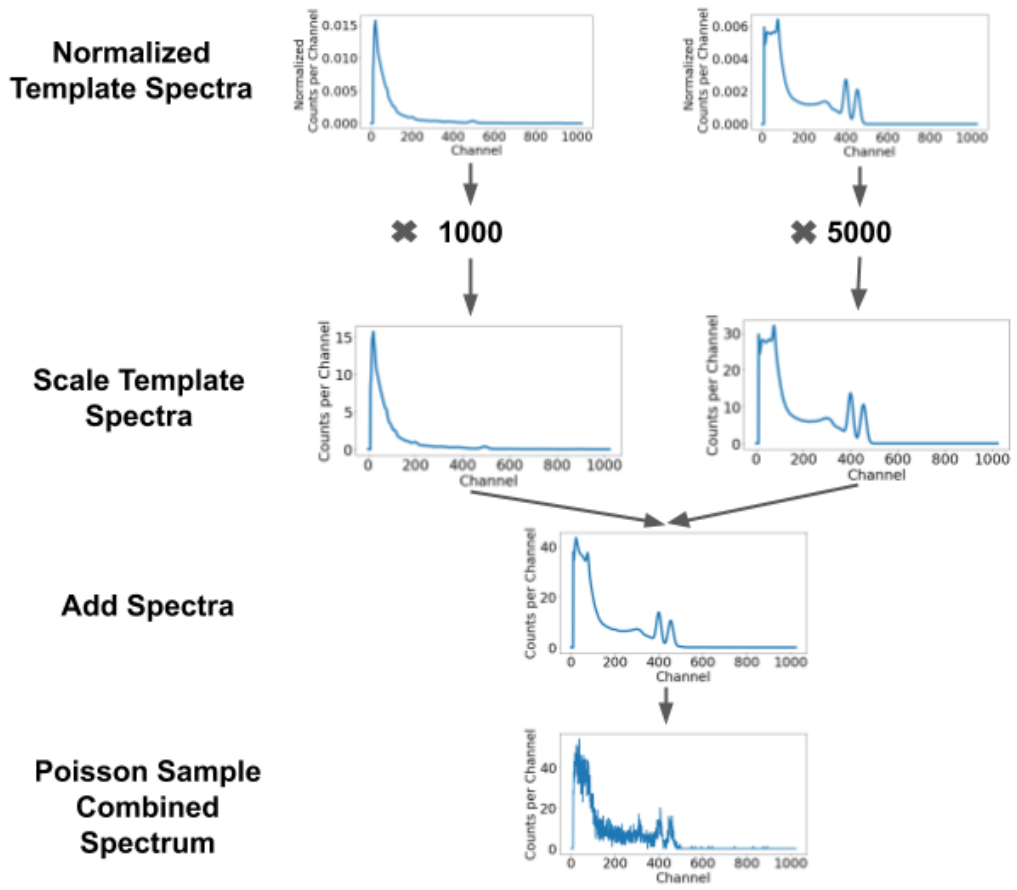


Figure 2.27: An example of the template sampling process combining ^{60}Co and background template spectra into a dataset entry.

learning training.

This workflow is designed to be flexible and allow for quick dataset prototyping. The workflow can be used for multiple gamma-ray transport and detector response software. Dataset simulation functions can be added or modified in `annsa` to extend its capabilities.

2.3.2 Machine Learning Models

`annsa` includes Python functions that build machine learning models using both low-level TensorFlow and the Keras `Sequential` API. Building functions using low-level TensorFlow allows for greater flexibility in model construction and training. The Keras `Sequential` API simplifies how models are trained and saved. `annsa` includes functions to build DNNs, 1D CNNs,

DAEs, and 1D CAEs using both of these methods. This section outlines the architectures available in `annsa`.

2.3.2.1 Dense Neural Network

DNNs in `annsa` are created by stacking dense-dropout layers between user-defined input and output vectors. An outline of this architecture is shown in Figure 2.28. The number of dense-dropout layers and the nodes in each layer are defined by a list of nodes provided by the user. The output length is also user-defined, allowing for the training of different sized isotope libraries for classification or training on a single regression target. The input vector length is also defined by the user. Additional necessary user-defined parameters are shown in Table 2.2.

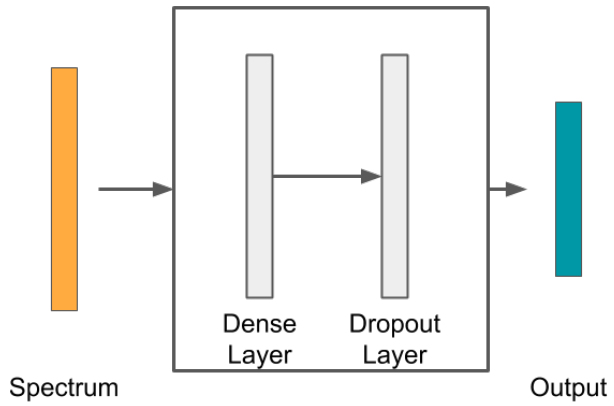


Figure 2.28: `annsa`'s DNN structure.

Table 2.2: `annsa`'s DNN parameters.

Parameter	**Default Value**
Dense Nodes per Layer	10
Dropout Rate	0.5
Activation Function	relu
L2 Regularization Strength	1 x 10^{-3}
Minibatch Size	32
Input Scaling	None
Output Size	30
Output Activation	softmax

2.3.2.2 Convolutional Neural Network

The convolution architecture included in `annsa` is based on a 1D convolution followed by a pooling operation. Two 1D convolution-pooling operations are illustrated in Figure 2.29. In the convolution, three filters are used. These filters will be referred to as blue, red, and green. Each filter is a vector of the same length, defined by the user. Each filter is convolved along the input and the result is placed in convolutional layer one. This process is illustrated for a single convolution using the red filter. Padding is included at the input signal edges to ensure the resulting convolution has the same length as the input. The pooling operation is indicated by a black box with a white border. Pooling is performed along the output of each filter. In this example max pooling is used with a stride of two. Max pooling takes the maximum value in the pooling length and places it in the next layer. By using a pooling stride of two, the filter is moved by two places before the pooling operation is performed, reducing the output size by a factor of two. Pooling in `annsa` includes zero padding by default to preserve the previous layer's size and to retain information at the signal's ends.

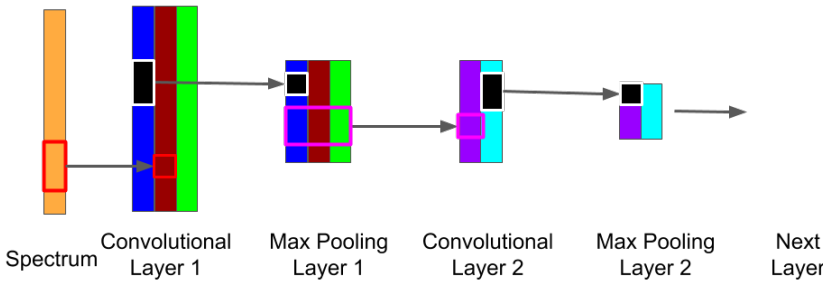


Figure 2.29: The 1D convolutional-pooling operation included in `annsa`.

Subsequent convolution-pooling layers also use 1D convolutions. In Figure 2.29, a second convolution operation is performed using purple and light blue filters. The filters used in each convolutional layer have a depth equal to the number of filters in the previous layer. In the example, this results in the second convolutional layer's filters having a depth of three. Each convolutional layer's filter lengths are mutually independent and defined by the user.

1D CNNs in `annsa` are created by stacking convolutional-pooling layers followed by dense-dropout layers, demonstrated in Figure 2.30. The final convolutional layer's output is flattened before being used by the dense layers. Parameters related to the convolutional architecture are shown in Table 2.3. Dropout and L2 regularization are only applied to the dense layers of this network.

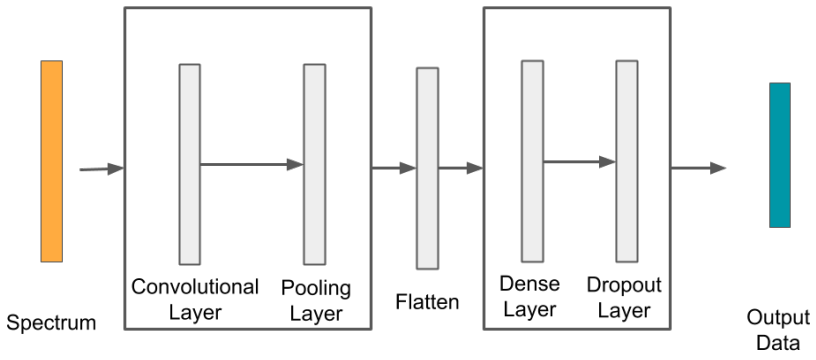


Figure 2.30: `annsa`'s CNN structure.

Table 2.3: `annsa`'s 1D CNN parameters.

Parameter	**Default Value**
Pooling Method	max pooling
Number of Filters per Layer	8
Length of Filter per Layer	8
Convolution Stride per Layer	1
Pooling Stride per Layer	2
Pooling Length per Layer	8
Dense Nodes per Layer	0.5
Dropout Rate	0.5
Activation Function	relu
L2 Regularization Strength	1 x 10^{-3}
Minibatch Size	32
Input Scaling	None
Output Size	30
Output Activation	softmax

2.3.2.3 Dense Autoencoder

`annsa` includes a class to create DAEs (dense autoencoders). The DAE class contains an encoder and decoder function. Both functions contain a user-defined number of dense-dropout layers and nodes in each layer. Using this, encoders and decoders with different dense architectures can be constructed. Parameters required by the DAE class are shown in Table 2.4. An example autoencoder with a two-hidden layer encoder, a single hidden-layer decoder, and no dropout is shown in Figure 2.31. By changing the input and output, autoencoders can be trained to extract various features from input signals. The final layer of the encoder, called the encoding, contains these extracted features. This example demonstrates a background-subtracting and denoising autoencoder: the input is a Poisson-sampled source and background spectrum and the output is only the source spectrum without Poisson sampling.

Weights from the encoder of a trained DAE can be loaded into a classification DNN for additional training in a process called fine tuning. This process is shown in Figure 2.32.

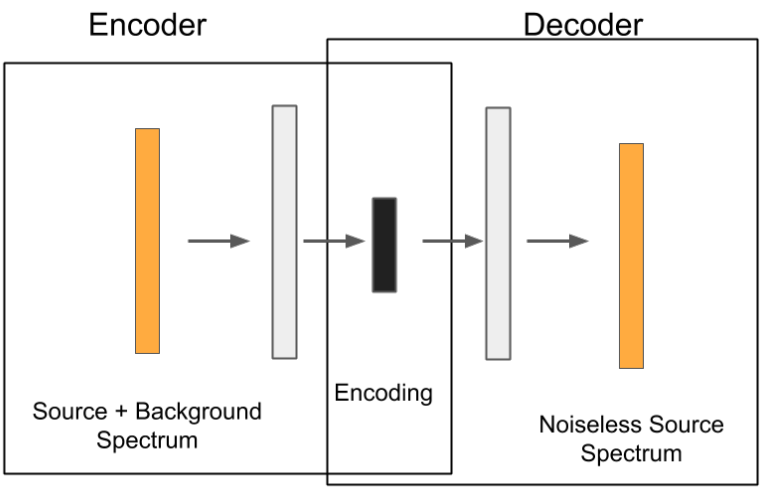


Figure 2.31: `annsa` DAE structure. This example demonstrates a background subtracting denoising autoencoder.

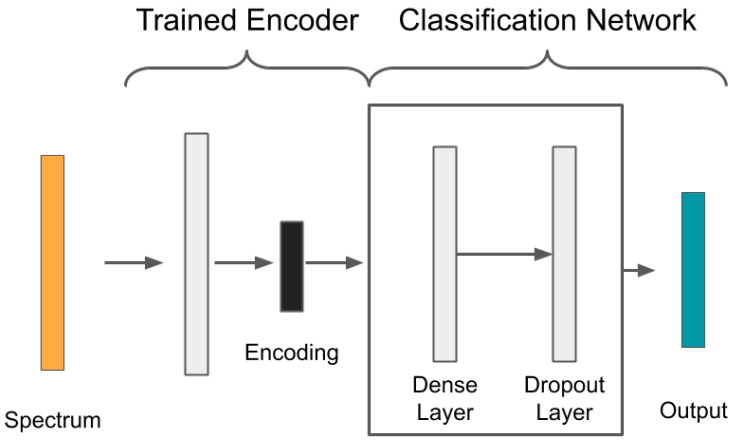


Figure 2.32: An example of a DAE's encoding being loaded into a classification DNN.

Table 2.4: `annsa`'s DAE parameters.

Parameter	**Default Value**
Dense Encoder Nodes	32
Dense Decoder Nodes	None
Dropout Rate	0.0
Activation Function	relu
Minibatch Size	32
Input Scaling	None
Output Size	1024
Output Activation	None

2.3.2.4 Convolutional Autoencoder

`annsa` includes a class to create CAEs. Like the DAE class, the CAE class contains an encoder and decoder function. The CAE uses parameters from the 1D CNN (Table 2.30) with the exception of L2 regularization strength and dropout rate (parameters only used by dense layers). The encoder is constructed using stacked 1D convolutional-pooling layers. As with the CNNs described in section 2.3.2.2, the CAEs use padding to keep the input and output sizes the same when the strides parameter is set to one. In `annsa`, the CAE encoder pooling strides parameter is two by default, making each convolutional-pooling layer reduce the input's length by a factor of two. The decoder uses resize-convolutional layers to increase the dimension of the encoding. The resize layer doubles the input signal's length using bilinear interpolation. The final decoding convolutional layer must have one output filter to ensure the autoencoder's output is a vector. A summary of this operation is shown in Figure 2.33.

Similar to the DAE, the weights from the encoder of a trained CAE can be extracted into a CNN for additional training. This process is shown in Figure 2.34.

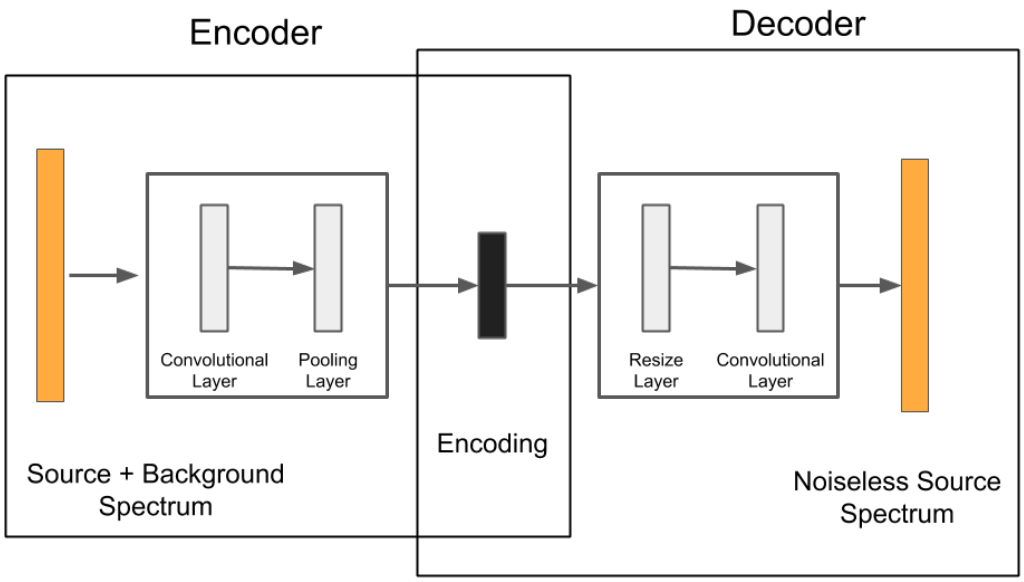


Figure 2.33: `annsa`'s CAE structure. This example demonstrates a background subtracting denoising autoencoder.

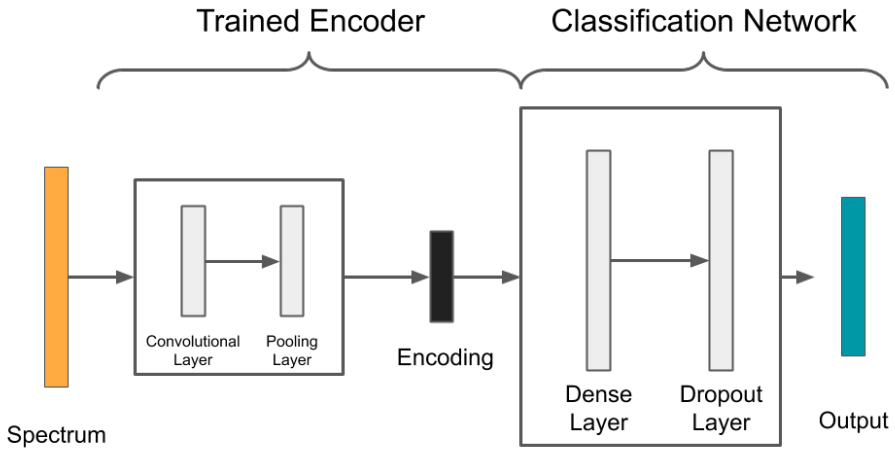


Figure 2.34: An example of a CAE's encoding being loaded into a CNN.

2.3.3 Training

Template datasets were simulated on a local machine and uploaded to a cloud computing platform using Amazon Web Services (AWS) as shown in Figure 2.35. Each model was trained using one of eight GPU's available on a p2.8xlarge EC2 (Elastic Compute Cloud) instance. Jupyter notebooks

used for training and generating results in this dissertation are available at `https://github.com/arfc/annsa/examples`.

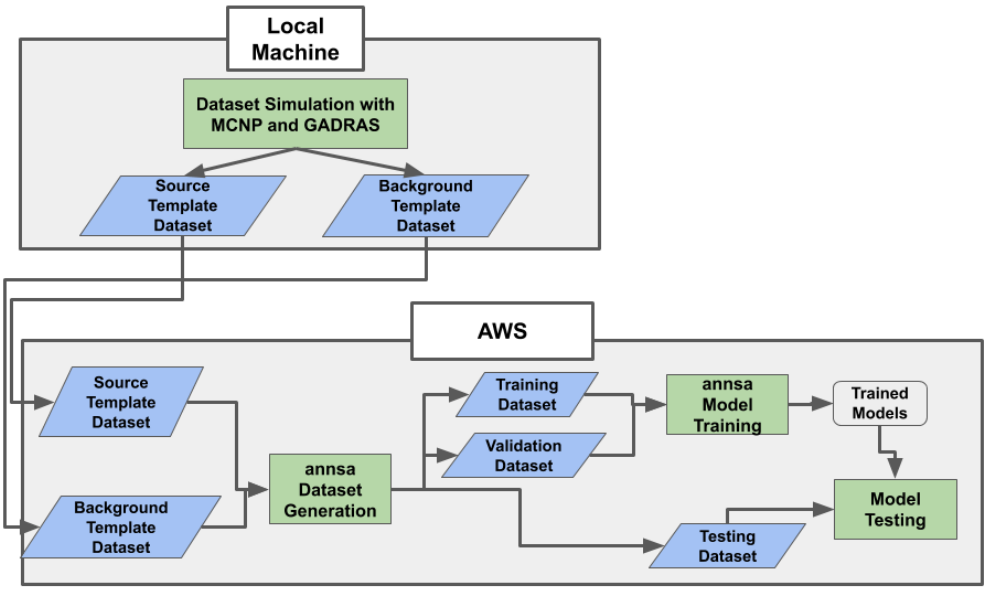


Figure 2.35: Diagram of AWS use.

2.4 Chapter Conclusions

In this chapter we described the physical processes responsible for signals in gamma-ray spectrometers and how multi-layer ANNs can be trained to solve diverse tasks. We also introduced `annsa`, the Python package we developed and used to create training datasets and train a variety of machine learning models for tasks in gamma-ray spectroscopy.

In the following chapters, ANNs will be presented using these concepts and their performance on a number of real and simulated spectra will be discussed.

CHAPTER 3

MACHINE LEARNING MODELS EXPLORED

In this chapter, we investigate which parameters affect the convolutional and dense models' performance for datasets of different complexity. Analyzing which hyperparameters most influence performance informs future research into applying machine learning to gamma-ray spectroscopy. To accomplish this, we ran hyperparameter searches using two datasets: a complete dataset with a wide range of simulated parameters and a simple dataset with smaller parameter ranges. This section describes these datasets and the architectural and regularization hyperparameters choices for the dense and convolutional models. This section concludes with a discussion on the insights obtained from the hyperparameter searches.

3.1 Training Templates Overview

The datasets used to train the models are created using templates of simulated gamma-ray spectra without Poisson noise. A one-dimensional particle transport code developed at Sandia National Laboratory, GADRAS-DRF (Gamma Detector Response and Analysis Software - Detector Response Function) [52], was used to generate these templates. This simulation code is used to model a radiation detector's response considering environmental scattering and a specific detector's properties. The GADRAS-DRF interface in Figure 3.1 shows the detector parameters used in this work. These parameters are from an Ortec 905-3 NaI(Tl) detector included in the Department of Homeland Security's Algorithm Improvement Program (AIP) software package [87]. To simulate our template spectra, the detector model's default energy calibration was modified so the detector measured energies from 0 MeV to 3.5 MeV. This was accomplished by setting the calibration offset (Order 0 in E) to zero and the calibration's gain (Order 1 in E) to 3500. The

default number of channels was also changed to 1194, calibrating the 1024^{th} channel - the default spectrum length used in this work - to an energy of 3 MeV. This ensures that isotopic signatures higher than 3 MeV are included in the training dataset when a template's calibration is modified to include these energies.

Figure 3.1: GADRAS-DRF GUI showing parameters used to simulate the Ortec 905-3 2x2-in NaI(Tl) detector used in this work.

GADRAS-DRF was also used to simulate spectral changes associated with measurement geometry. The measurement scenario used by GADRAS-DRF is illustrated in Figure 3.2. Using GADRAS-DRF, we can simulate changes in source-detector distance, the source and detector's mutual height from the ground, and optional shielding material between the source and detector. To demonstrate two of these effects, ^{60}Co spectra were simulated at various source-detector distances, Figure 3.3, and heights above the ground, Figure 3.4. These parameters change the Compton continuum's shape, the peak-to-total ratio, and the backscatter peak's magnitude. Another parameter that changes the spectrum is the Gaussian energy broadening of the photopeaks. Due to manufacturing differences, each detector has a different amount of energy broadening. Examples of ^{60}Co spectra with different FWHM parameters are shown in Figure 3.5.

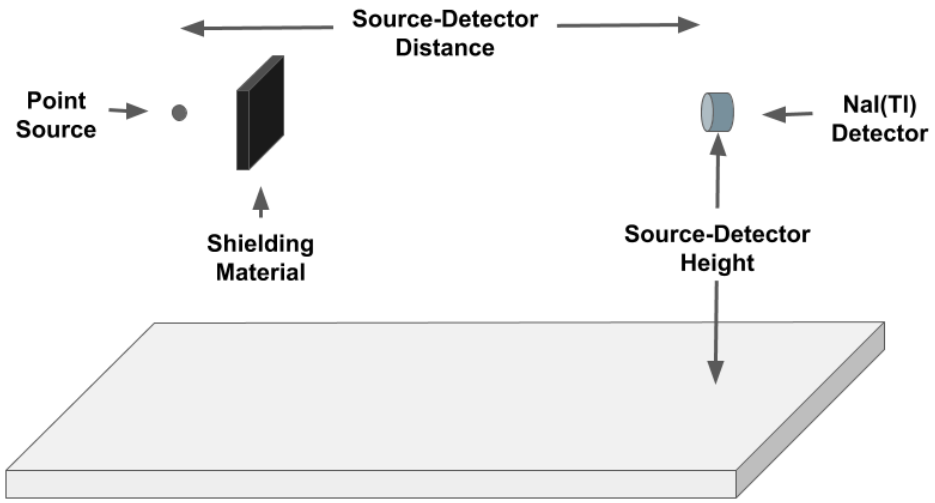


Figure 3.2: GADRAS-DRF measurement diagram. Environmental scatter is approximated using the photon scatter terms from Figure 3.1.

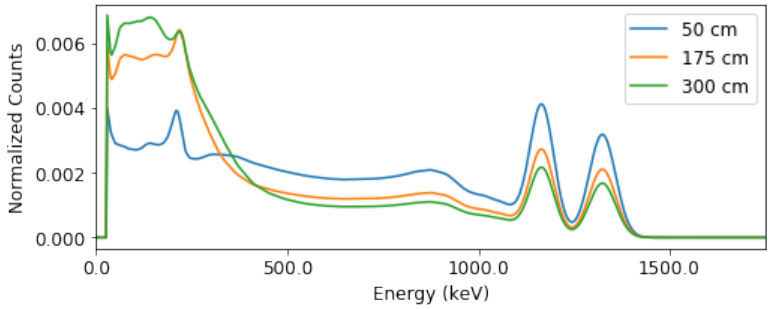


Figure 3.3: Comparison of a ^{60}Co spectrum simulated using various source-detector distances.

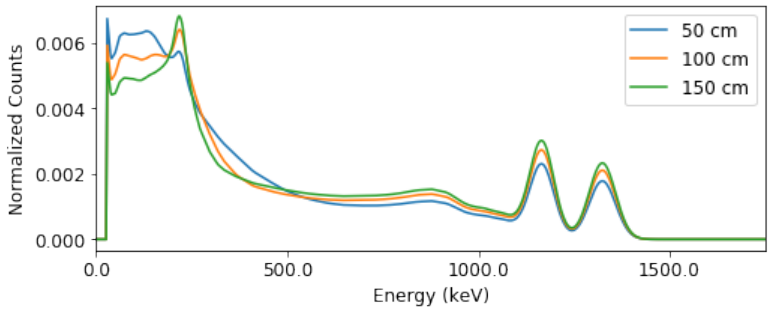


Figure 3.4: Comparison of a ^{60}Co spectrum simulated using various source-detector heights off the ground.

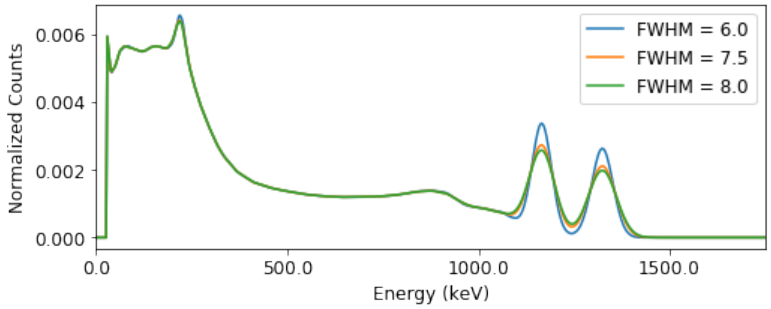


Figure 3.5: Comparison of a ^{60}Co spectrum simulated with various FWHM parameters.

The template parameters used in this study are shown in Table 3.1. These parameters are based on handheld RIID scenarios. The source-detector height off the ground are sampled between shin and arm heights (50 cm to 150 cm). The source-detector distance range corresponds to standoff distances expected when measuring a source in a cargo container. The areal density of each material corresponds to 20%, 40%, 60%, and 80% attenuation of a 200 keV photon. This photon energy was chosen because it is near the 186 keV energy of the characteristic ^{235}U photopeak. A set of unshielded templates is also included. The FWHM range was chosen based on reported FWHM values for a NaI(Tl) detector (see section 2.1.2). Background locations were chosen based on their geographic and geologic differences.

Table 3.1: Fixed GADRAS-DRF template simulation parameters.

Simulation Parameter	**Values**
Source-detector height [cm]	50, 100, 125
Source-detector distance [cm]	50, 175, 300
Areal density of solid aluminum [$\frac{g}{cm^2}$]	1.82, 4.18, 7.49, 13.16
Areal density of solid iron [$\frac{g}{cm^2}$]	1.53, 3.5, 6.28, 11.02
Areal density of solid lead [$\frac{g}{cm^2}$]	0.22, 0.51, 0.92, 1.61
FWHM at 662 keV [%]	7.0, 7.5, 8.0
Background Location	Albuquerque, Atlanta, Austin, Chicago , Knoxville , Miami

In addition to the parameters simulated by GADRAS-DRF, parameters are included to perform additional data augmentation. These parameters are shown in Table 3.2.

Table 3.2: Default data augmentation parameters.

Simulation Parameter
integration time [t]
Background Count Rate [cps]
Signal to Background ratio
Calibration - Offset [channels]
Calibration - Gain

3.2 Hyperparameter Search

To determine which parameters affect the convolutional and dense model's performance on datasets of increasing complexity, hyperparameter searches for each model are performed for a simple and more complicated dataset. The hyperparameter search for a single model and dataset is illustrated in Figure 3.7. For each dataset and model, 5-folds cross validation is performed on 256 sets of randomly chosen hyperparameters. The maximum number of epochs was set to 200 and an early stopping patience of 20 epochs was used to stop training with the validation dataset's F1 score. In addition to this, training was ended after 10 epochs if the validation dataset's F1 score did

cross a 10% threshold. The hyperparameter combination with the lowest average validation dataset F1 score was declared the optimum hyperparameter combination for that model. The validation dataset being used to end training induces a bias into the score used to determine the best hyperparameter combination. To remove this bias, a separate testing dataset's F1 score would need to be used to determine the optimum hyperparameter combination.

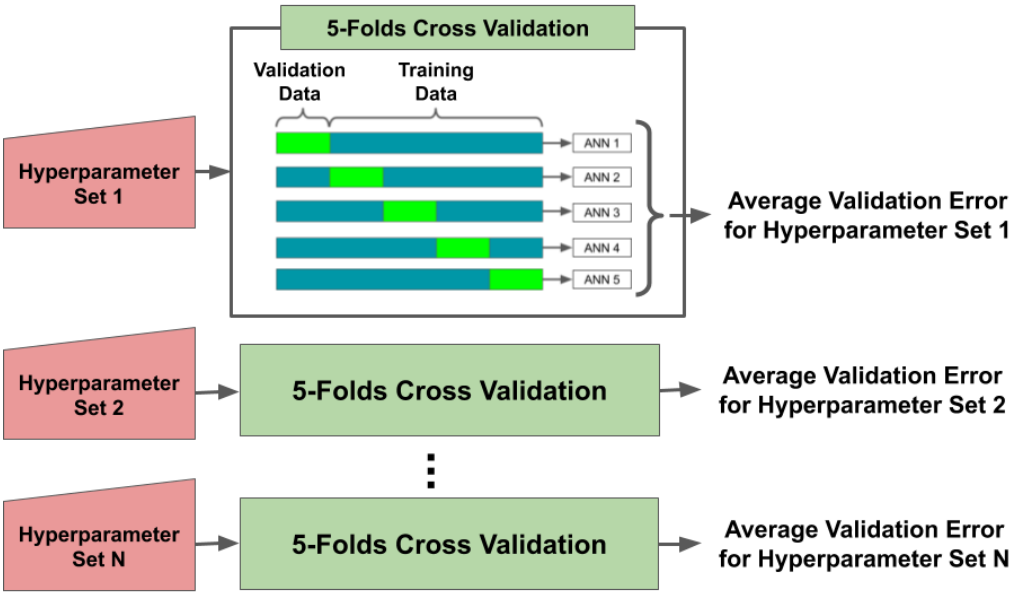


Figure 3.6: Hyperparameter search workflow.

3.3 Datasets Used for the Hyperparameter Search

The parameters used for the simple dataset are shown in Table 3.3 and parameters used for the complete dataset are shown in Table 3.4. Datasets are simulated using the process described in Section 2.3.1. Isotopes included in the dataset are from the ANSI N42-34-2006 standard for isotope identification devices [88]: ^{241}Am, ^{133}Ba, ^{57}Co, ^{60}Co, ^{51}Cr, ^{137}Cs, ^{152}Eu, ^{67}Ga, ^{123}I, ^{125}I, ^{131}I, ^{111}In, ^{192}Ir, ^{177m}Lu, ^{99}Mo, ^{237}Np, ^{103}Pd, ^{239}Pu, ^{240}Pu, ^{226}Ra, ^{75}Se, ^{153}Sm, ^{99m}Tc, ^{201}Tl, ^{204}Tl, ^{233}U, ^{235}U, ^{238}U, and ^{133}Xe. In each dataset, a total of 100 spectra were simulated for each isotope. Each dataset also included 100 spectra of only background.

Table 3.3: Range of parameters used for the simple dataset.

Simulation Parameter	**Parameter Range**	**Sampling**
Source-Detector Distance [cm]	175	N/A
Source-Detector Height [cm]	100	N/A
FWHM at 662 keV [%]	7.5	N/A
Shielding [% 200 keV Attenuated]	0%, 20%	Uniform
Integration Time [s]	60 - 600	Log-Uniform
Calibration - Offset [channels]	0 - 10	Uniform
Calibration - Gain	0.9 - 1.1	Uniform
Signal to Background Ratio	0.5 - 2.0	Uniform
Background Count Rate [cps]	200	Poisson

Table 3.4: Range of parameters used for the complete dataset.

Simulation Parameter	**Parameter Range**	**Sampling**
Source-Detector Distance [cm]	50, 175, 300	Uniform
Source-Detector Height [cm]	50, 100, 150	Uniform
FWHM 662 keV [%]	7.0, 7.5, 8.0	Uniform
Shielding [% 200 keV Attenuated]	0%, 20%, 40%, 60%	Uniform
Integration Time [s]	10 - 3600	Log-Uniform
Calibration - Offset [channels]	0 - 10	Uniform
Calibration - Gain	0.8 - 1.2	Uniform
Signal to Background Ratio	0.1 - 3.0	Uniform
Background Count Rate [cps]	200	Poisson

The hyperparameter search for a single model and dataset is illustrated in Figure 3.7. For each dataset and model, 5-folds cross validation is performed on 256 sets of randomly chosen hyperparameters. The maximum number of epochs was set to 200 and an early stopping patience of 20 epochs was used to stop training using the validation dataset's F1 score,

$$\text{F1 score} = 2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{3.1}$$

where

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$
$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}.$$

In addition to this, training was ended after 10 epochs if the validation dataset's F1 score did cross a 10% threshold. The hyperparameter combination with the lowest average validation dataset F1 score was declared the optimum hyperparameter combination for that model. The validation dataset being used to end training induces a bias into the score used to determine the best hyperparameter combination. A better approach would be to use a separate simulated dataset's F1 score to determine the optimum hyperparameter combination.

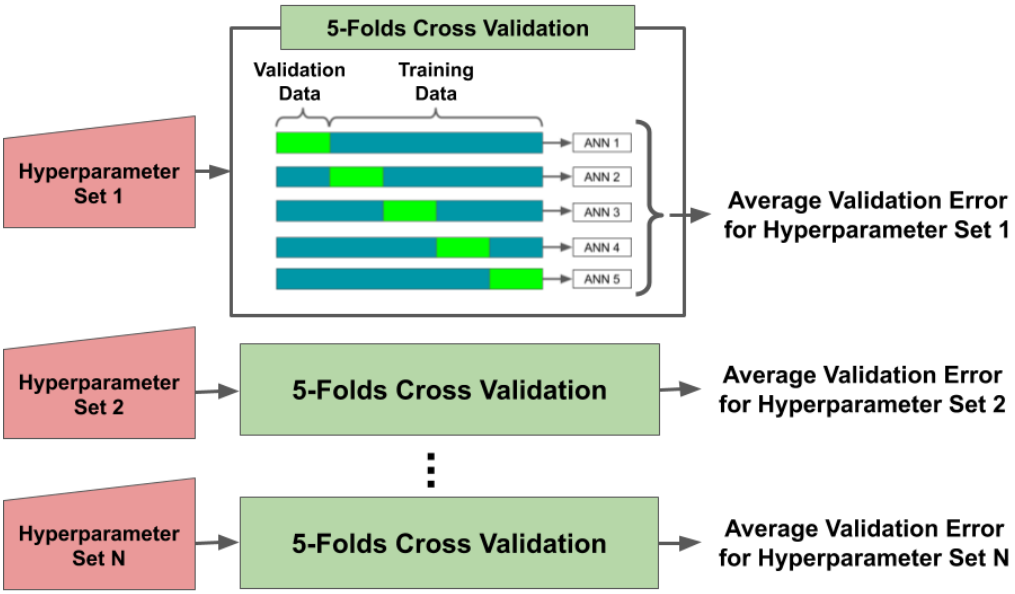


Figure 3.7: Hyperparameter search workflow.

3.4 Hyperparameter Search Results

In this section, hyperparameter search results are shown using random efficiency curves and by comparing parameter values versus average validation set F1 scores from the 5-fold cross validation. Random efficiency experiment

curves indicate the quality of the hyperparameter search space and allow for reproducibility. Analyzing how the parameter values change the F1 score indicates which parameters are important. Hyperparameter bounds are based on previous published experiments as well as literature recommendations [8, 18, 89].

3.4.1 Dense Architecture

The architecture and training hyperparameters used to construct DNN's are shown in Table 3.5. The number of densely connected nodes decreases for each subsequent layer. The input scaling is read left-to-right. For example, the $sqrt - max$ scaling would first take the square root of the each channel in the spectrum and then normalized the spectrum by its maximum value. The $L1 - norm$ normalizes a spectrum by its L1 norm. The $log1p$ function is defined as

$$log1p(x) := log_{10}(x + 1). \tag{3.2}$$

Table 3.5: Range of hyperparameters explored for the DNN.

Hyperparameter	**Hyperparameter Range**	**Sampling**
Number of Layers	1 - 3	Uniform
Nodes in Layer	32, 64, 128, 256, 512	Uniform
Initial Learning Rate	10^{-4} - 10^{-1}	Log-Uniform
L2 Regularization Strength	10^{-2} - 10^{0}	Log-Uniform
Dropout Frequency	0 - 1	Uniform
Batch Size	16, 32, 64, 128, 256, 512	Uniform
Activation Function	relu, tanh	Uniform
Input Scaling	sqrt sqrt-max sqrt-L1 norm log1p log1p-max log1p-L1 norm	Uniform

Random efficiency experiment curves for the DNN trained on the simple

and complete datasets are shown in Figures 3.8 and 3.9. A large number of hyperparameter combinations did not reduce the validation dataset's F1 score above the 10% threshold after 10 epochs, ending their training early. As expected, the complete dataset took more trials to obtain good performance. The median validation dataset's F1 score in the simple dataset begins to asymptote after an experiment size of 16 while the validation dataset's F1 score in the complete dataset does not asymptote. This indicated that when applying DNN's to more difficult problems in gamma-ray spectroscopy either wider hyperparameter ranges need to be explored, more advanced hyperparameter search strategies need to be employed, or that due to their structure DNN's are not well suited to perform tasks in gamma-ray spectroscopy.

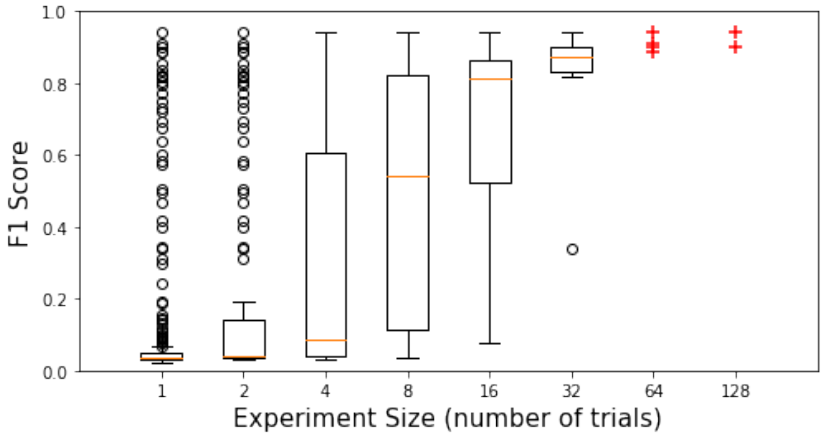


Figure 3.8: Random hyperparameter search efficiency curves for the DNN using the simple dataset. Red crosses indicate individual experiments.

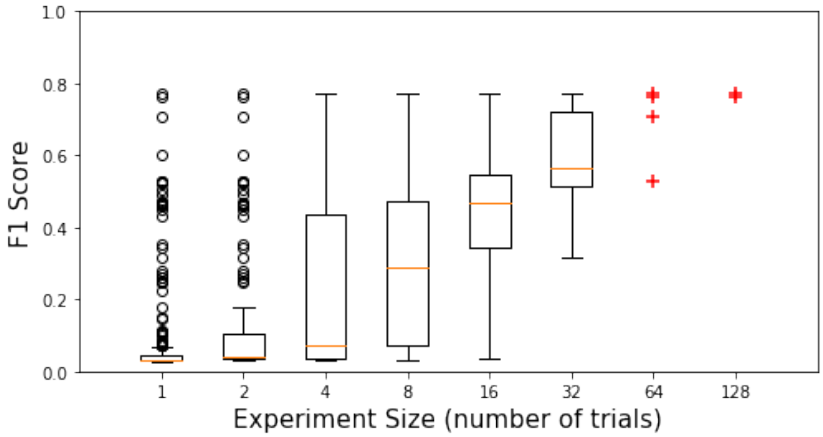


Figure 3.9: Random hyperparameter search efficiency curves for the DNN using the complete dataset. Red crosses indicate individual experiments.

Figure 3.10 shows the distribution of average validation dataset F1 score from the 5-folds cross validation for each hyperparameter. Conclusions based on these will suffer from a small sample size because few networks trained on the complete dataset achieved high validation dataset F1 scores. General trends in the simple dataset are more representative of actual hyperparameter performance compared to the complete dataset.

Sub-figure 3.10a shows that a learning rate less than 10^{-2} should be used when conducting hyperparameter searches for spectroscopic datasets of similar complexity. This figure also shows that the complete dataset prefers a slower learning rate, with best performing networks using learning rates below $10^{-3.5}$. Smaller learning rates should also be explored in future searches, if computationally feasible.

Sub-figure 3.10b shows that the dropout rate has a less obvious performance cutoff compared to the learning rate. This is likely because the dropout value does not have a large effect on training.

Sub-figure 3.10c shows that the simple dataset works well in a variety of mini-batch sizes, while the complete dataset may require a larger mini-batch size.

Sub-figures 3.10f and 3.10d show the effect of model capacity on performance. Model capacity is comparable to the number of free parameters of a model; too much capacity can lead to overfitting and too little capacity may be insufficient to fit complex data. Overfitting can be seen in the performance drop in three dense layers. Additional layers are unnecessary for problems of this difficulty with the hyperparameters explored. For the complete dataset a significant number of networks perform well with a total number of nodes between 250 and 1000. The simple dataset performed well with a wide range of total nodes.

Sub-figures 3.10g shows the effect of feature preprocessing on performance. L1 normalization performs very poorly due to the numerical scale of the features. Gradients computed with learning rates explored by the hyperparameter search are too small to significantly update the weights because the features are on the order 10^{-3}. Features explored by the other methods are typically between 10^{0} - 10^{2}. Scaling spectra using the *sqrt* of their features makes networks perform well for both datasets. In particular, using $sqrt - max$ scaling produces the most models with high validation dataset F1 scores.

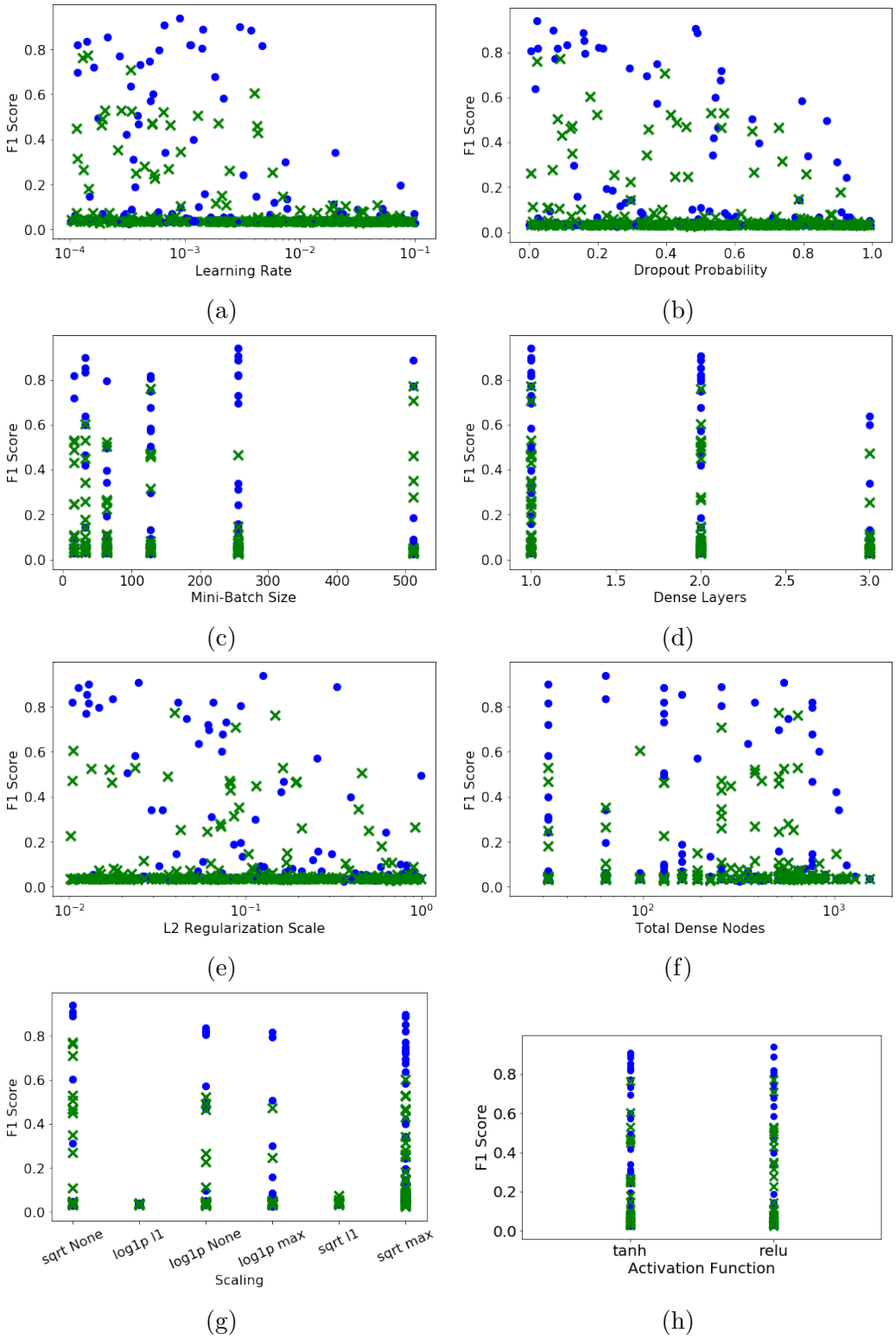


Figure 3.10: Effect of dense hyperparameters on the validation dataset's final F1 score.

Table 3.6: Optimum DNN hyperparameter combination found for the simple and complete dataset.

Hyperparameter	**Optimum Value**	
	Simple Dataset	**Complete Dataset**
Dense Layer Hidden Nodes	64	32
Initial Learning Rate	0.00089	0.00024
L2 Regularization Strength	0.13	0.0097
Dropout Frequency	0.023	0.16
Batch Size	256	16
Activation Function	relu	relu
Input Scaling	sqrt	sqrt-max

3.4.2 Convolution Architecture

The architecture and training hyperparameters used to construct CNN's are shown in Table 3.7. Note, as with the DNN the number of densely connected nodes decreased with each subsequent layer. A smaller range of batch sizes was searched in the CNN compared to the DNN due to computational constraints.

Table 3.7: Range of hyperparameters explored for the CNN.

Hyperparameter	**Hyperparameter Range**	**Sampling**
Filter Kernels in Each Layer	4	Uniform
	8	
	16	
	32	
	4 - 8	
	8 - 16	
	16 - 32	
	4 - 8 - 16	
	8 - 16 - 32	
Filter Kernel Lengths	2, 4, 8, 16	Uniform
Pooling size	2, 4, 8, 16	Uniform
Number of Dense Layers	1 - 3	Uniform
Nodes in Dense Layers	10 - 1000	Log-Uniform
Initial Learning Rate	10^{-4} - 10^{-1}	Log-Uniform
L2 Regularization Strength	10^{-3} - 10^{0}	Log-Uniform
Dropout Frequency	0 - 1	Uniform
Batch Size	16, 32	Uniform
Activation Function	tanh, relu	Uniform
Input Scaling	sqrt	Uniform
	sqrt-max	
	sqrt-L1 norm	
	log1p	
	log1p-max	
	log1p-L1 norm	

Random efficiency experiment curves for the CNN trained on the simple and complete datasets are shown in Figures 3.11 and 3.12. Similar to the dense network, a large number of hyperparameter combinations did not reduce their validation dataset's F1 score in the training set in enough epochs and their training was ended early. Contrary to the DNN, the median validation dataset's F1 score in both datasets smoothly increased with additional trials. Both datasets achieved validation dataset F1 scores above 90%. This demonstrates that the hyperparameter bounds used for both problems are

well suited to the problems and that the CNN architectures explored in this search have more potential than the DNN architectures for gamma-ray spectroscopy. As with the DNN, validation dataset F1 scores on the simple dataset are higher than the validation dataset F1 scores on the complete dataset.

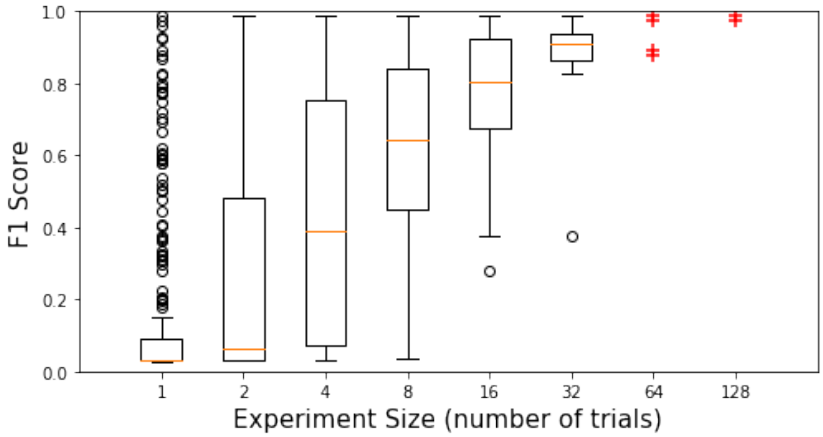


Figure 3.11: Random hyperparameter search efficiency curves for the CNN using the simple dataset. Red crosses indicate individual experiments.

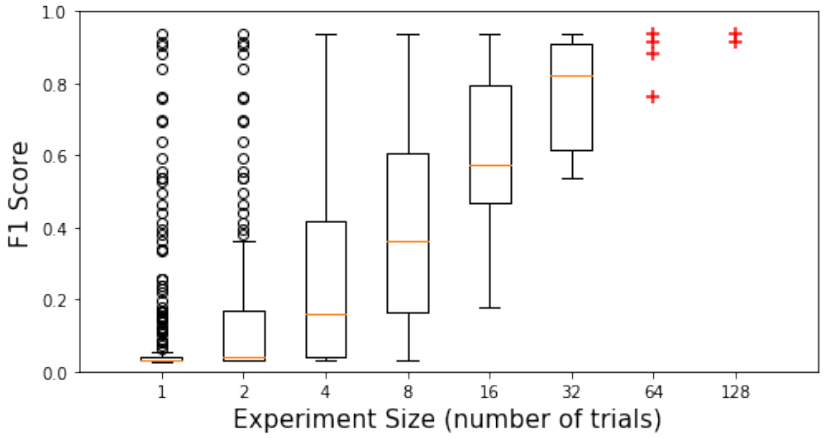


Figure 3.12: Random hyperparameter search efficiency curves for the CNN using the complete dataset. Red crosses indicate individual experiments.

Figure 3.13 shows the distribution of hyperparameters searched for the CNN and their average validation dataset F1 scores from the 5-folds cross validation. Similar to the DNN, few networks trained on the complete dataset achieved high validation dataset F1 scores. Conclusions based on these diagrams suffer from small (albeit larger than the DNN) sample sizes.

The CNN's training performance is largely agnostic to many hyperparameters, including those associated with the dense part of the CNN. Figure 3.13 shows that both datasets train similarly over a wide range of learning rates below 10^{-2}, dense layer dropout rates below 0.8, L2 regularization scales below 10^{-1}, between 50 - 200 total dense nodes, and both mini-batch sizes.

The CNN's training performance on the complete dataset is sensitive to hyperparameters related to model capacity. The total number of convolutional layers have the largest affect on the performance of models trained using the complete dataset. Because of the additional complexity in the complete dataset, deeper networks - which extract more abstract features - are necessary for good performance. CNNs with additional convolutional layers should be explored for problems of similar complexity. Additional dense layers did not provide the same boost in performance. Models trained using the simple dataset perform well with a large range of total dense layers while the complete dataset performs well with fewer layers, achieving the best validation dataset F1 score with a two layers. Simple and complete models with three dense layers have maximum F1 scores lower than two or three dense layers.

Performance is also sensitive to other convolutional hyperparameters. Models trained using the simple datasets perform well with each convolutional pooling size. As a general trend, models trained with the complete dataset perform better with larger convolutional pooling sizes. Longer pooling sizes add more shift and scale invariance, which is more important in the complete dataset due to a wider range of detector gain settings. Longer pooling sizes are necessary to incorporate photopeaks and Compton continua, where smaller pooling sizes may be sufficient to retain only photopeak information. Identifying the photopeaks may be enough to identify isotopes in the simple dataset because there are fewer confounding variables. Convolutional kernel lengths of 16 are required for optimal performance for both datasets. The preference for depth in the convolutional part of the CNN is also seen in the number of convolutional filters.

The effect of scaling on performance was similar to the DNN. Using *sqrt* scaling yielded the best performing networks, especially when tied with *max* normalization.

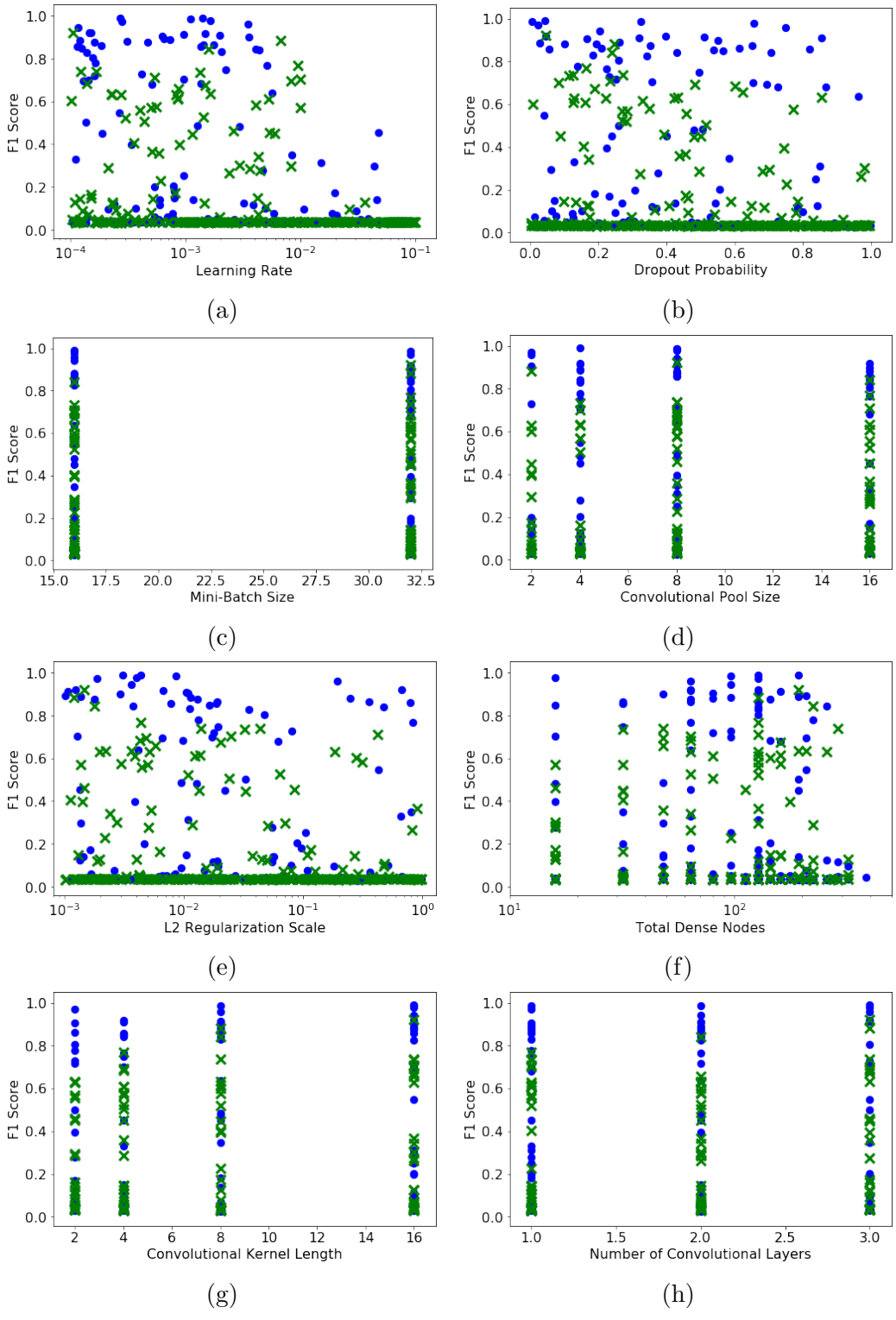


Figure 3.13: Effect of CNN hyperparameters on the validation dataset's final F1 score.

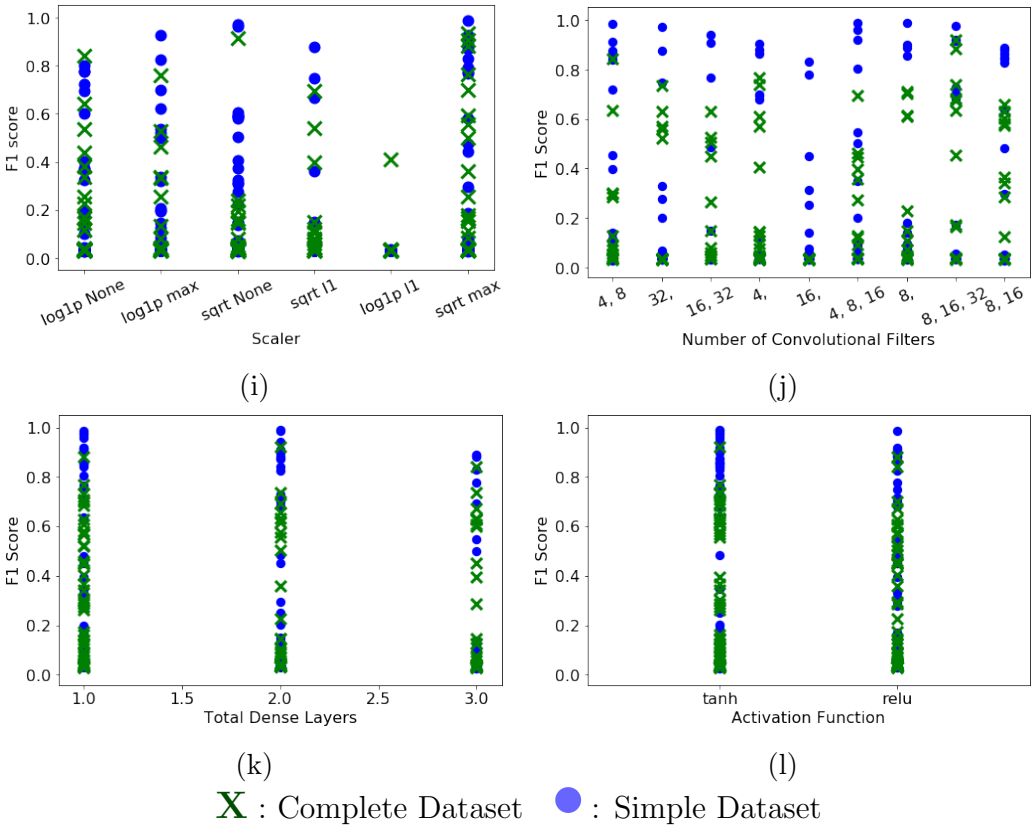


Figure 3.13 (cont.): Effect of CNN hyperparameters on the validation dataset's final F1 score.

Table 3.8: Optimum CNN hyperparameter combination found for the simple and complete dataset.

Hyperparameter	**Optimum Value**	
	Simple Dataset	**Complete Dataset**
Filter Kernels in Each Layer	4 - 8 - 16	8 - 16 - 32
Filter Kernel Length	16	16
Pooling size	4	8
Dense Layer Structure	128 - 64	128 - 64
Initial Learning Rate	0.0014	0.00010
L2 Regularization Strength	0.0044	0.0015
Dropout Frequency	0.042	0.045
Batch Size	16	32
Activation Function	tanh	tanh
Input Scaling	sqrt-max	sqrt-max

3.4.3 Autoencoder Architectures

Without an autoencoder, a single ANN has to learn multiple tasks to identify isotopes. An ANN would have to simultaneously identify the detector calibration, background signal, and source signal. By pretraining each network using an autoencoder to reconstruct a background-subtracted spectrum, the task of isotope identification is simplified for the ANN. To determine if using pretrained models results in more accurate identifications, a DAE and CAE were trained using the model parameters found in the hyperparameter search.

Pretraining was performed using a dataset of 10000 samples generated using the simple and complete dataset parameters. Because of the implicit regularization of the undercomplete encoding and the denoising background subtracting process, neither network uses additional regularization when training. The initial learning rate of each autoencoder was set to 10^{-5}.

CHAPTER 4

URBAN SOURCE IDENTIFICATION RESULTS AND DISCUSSION

This chapter applies machine learning algorithms to solve the problem of identifying a radioactive source in an urban environment. This scenario is applicable when performing source interdiction searching cargo containers, vehicles at boarder crossings, or surveying high profile events. Urban environments present unique challenges to gamma-ray spectroscopy. Background radiation can change over city blocks due to different concentrations of uranium and thorium in building materials. Sources may be purposely shielded by unknown amounts of material to obscure their gamma-ray signal.

To investigate how simulated parameters affect the performance of dense and convolutional networks with and without autoencoder pretraining, we first determine a sufficient training dataset size for each model and dataset by analyzing their learning curves. We then observe how well each model identifies simulated and measured spectra in a variety of conditions. Finally, we compare the F1 score from our approach with a peak-based Bayesian classifier.

4.1 Learning Curve Analysis

In this section we analyze learning curves to determine a sufficient number of training dataset examples for the simple and complete networks. Two separate datasets were constructed for this chapter: one using simple dataset parameters, Table 3.3, and the other using complete dataset parameters from Table 3.4. The datasets included isotopes from the ANSI N42-34-2006 standard, shown in Section 3.3. Each dataset contained 1 x 10^4 randomly generated spectra for each isotope. These datasets will be referred to as the simple master dataset and complete master dataset. Models trained using subsets of the simple and complete datasets are referred to as simple and complete

models.

Different sized training datasets were sampled from both master datasets and used to train each model. This process is illustrated in Figure 4.1. Stratified sampling without replacement was used to sample each master dataset. Stratified sampling keeps the number of isotopes in each sampled subset as equal as possible. Training dataset sizes included 50, 100, 500, 1000, 5000, 10000, 15000, and 20000. For each training dataset size, five sampled subsets of a master dataset were used as training datasets. ANNs were trained using these datasets. A separate holdout validation dataset of 300 spectra (10 examples per class) was used to end training when its F1 score did not improve using an early stopping patience of 10 epochs.

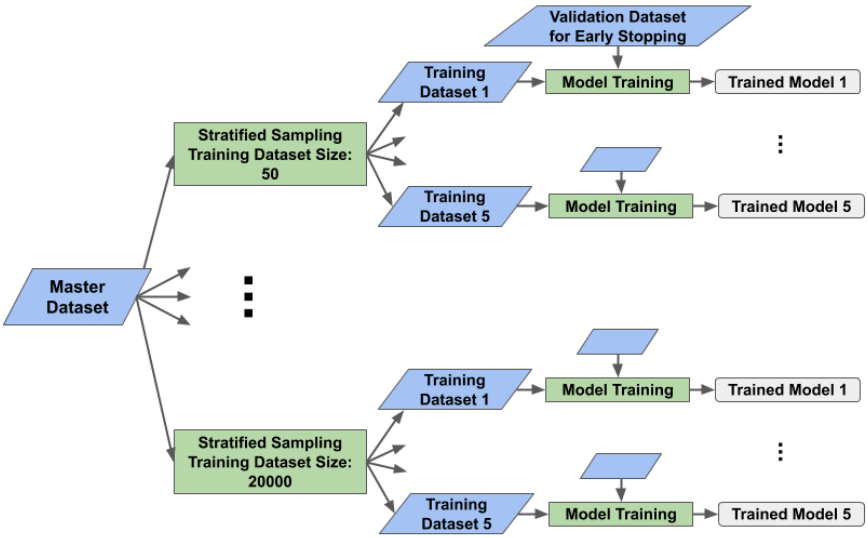


Figure 4.1: Training process used to train models for the learning curves.

The validation dataset's F1 score was used to construct learning curves for each model. Figure 4.2 shows the learning curves for the simple convolutional model's F1 scores reaching an asymptote near one above datasets of 5000 examples. This shows that each convolutional model is well suited to generalize within the simple parameter space. There are no obvious differences between the learning curves for the simple CAE and CNN.

Simple dense models reach an asymptote with a wider variance between F1 scores of 0.8 and 0.9. With fewer than 5000 training examples, the simple DAE had achieved F1 scores lower than the other models. This indicates that the simple DAE did not learn features from the pretraining step. This is

likely due to the dense network having an insufficient capacity to reconstruct spectra.

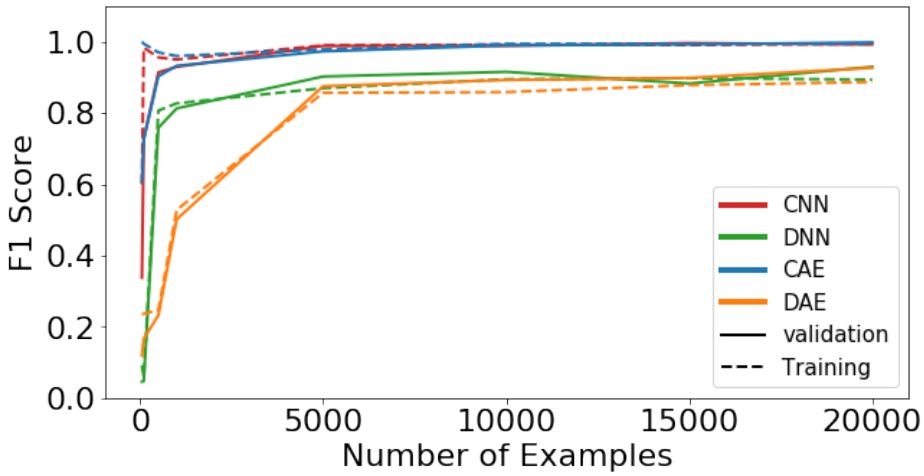


Figure 4.2: Learning curves for each simple model.

Table 4.1: Normalized area under the learning curves generated using the simple training dataset.

	Training	**Validation**
DNN	0.87	0.88
DAE	0.81	0.82
CNN	0.98	0.98
CAE	0.98	0.97

The validation dataset error curves in Figure 4.3 are less step than those in the simple learning curve plot. The asymptote for the full dense and convolutional models is also lower than the asymptote for the easy models, demonstrating that the complete dataset is more difficult for the models to generalize to. Even with a comparable number of trainable parameters, the convolutional networks outperform the dense networks. This indicates that the convolutional architecture is better suited to performing gamma-ray spectroscopy.

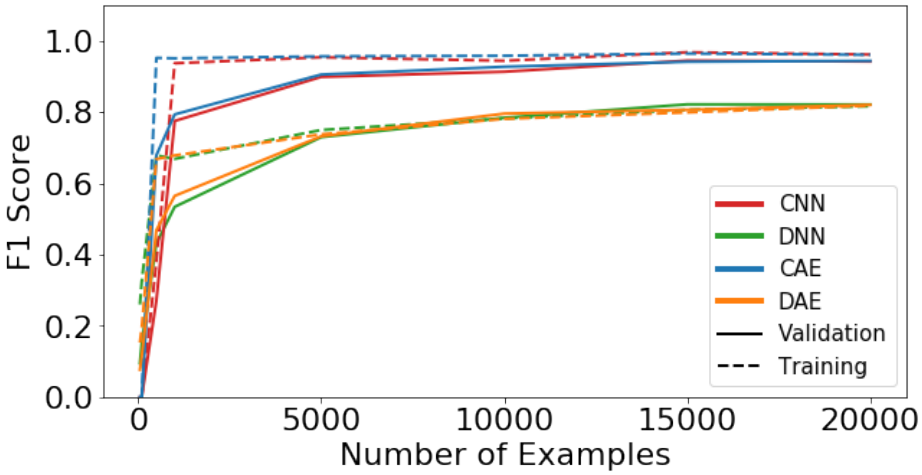


Figure 4.3: Learning curves for each complete model.

Table 4.2: Normalized area under the learning curves generated using the complete training dataset.

	Training	**Validation**
DNN	0.76	0.74
DAE	0.76	0.74
CNN	0.93	0.88
CAE	0.94	0.89

4.2 Performance in Simulated Datasets

In this section we analyze each simple and complete ANN's performance dependence on simulated parameters expected during source interdiction measurements. The process used in this section is outlined in Figure 4.4. For each ANN, five models are used as an ensemble by averaging their outputs. Each output is a posterior probability distribution over each isotope in the training library. Models trained using a dataset of 10000 examples from Section 4.1 were used in this section.

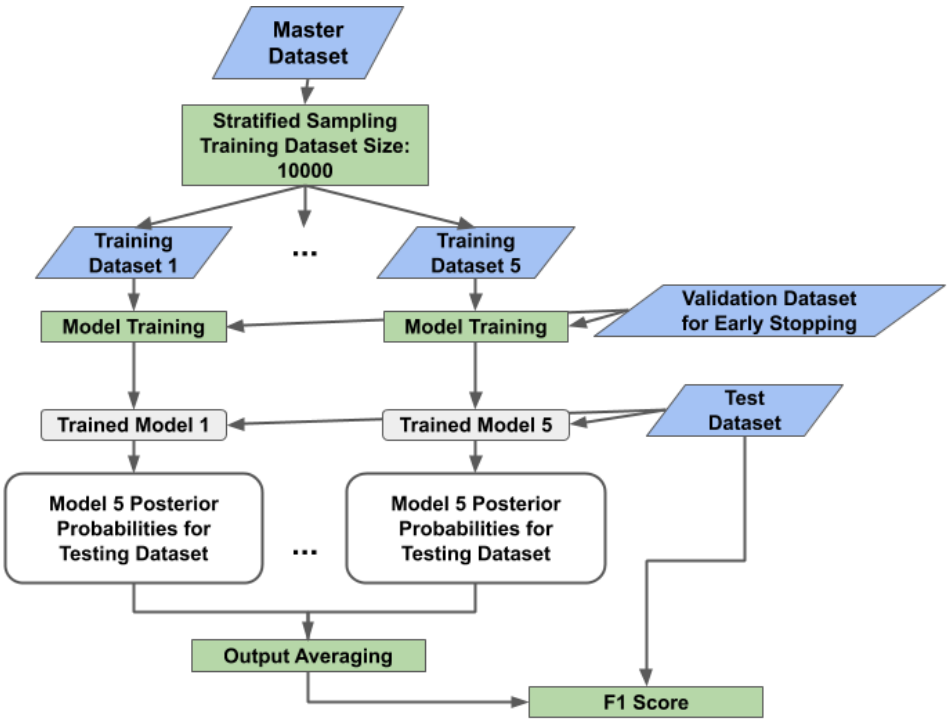


Figure 4.4: Model averaging process used to evaluate ANNs using testing datasets.

To compare each ANN's performance, we simulated datasets with various source-detector heights, source-detector distances, detector resolutions, shielding thicknesses, and detector gain calibrations. Datasets were simulated with ten spectra for each ANSI N42-34-2006 isotope and unique parameter. Each spectrum contains a single isotope and background. Default simulation parameters for each testing dataset are shown in Table 4.3. Testing datasets were simulated over integration times log-uniformly sampled from one second to an hour and signal to background ratios of 0.1 and 0.5.

Table 4.3: Default parameters used for all testing datasets.

Simulation Parameter	**Value**
Source-Detector Distance [cm]	175.0
Source-Detector Height [cm]	100.0
FWHM 662 [keV]	7.0
Shielding [% 200 keV Attenuated]	0%
Calibration - Offset [channels]	0.0
Calibration - Gain	1.0
Background Counts Per Second	200.0

4.2.1 Generalization Performance Dependence on Source-Detector Height

In this section we analyze how each model's F1 score changes when identifying spectra simulated with different source-detector heights off the ground. This tests if each model is sensitive to changes in the Compton continuum due to environmental scattering associated with the source-detector height. An example of these changes can be seen in Figure 3.4. These changes are only noticeably significant in the region below the backscatter peak around 200 keV. This region is important because it contains photopeaks used to detect enriched uranium.

As seen in Figures 4.5, 4.6, and 4.7, the performance of each network is insensitive to changes in source-detector height. This shows that performance is not impacted by the relatively small changes in the continuum caused by varying the source-detector height. Figure 4.5 also shows that simple models generalize to changes in source-detector height when the signal-to-background ratio is raised. This shows that the features required to accurately identify low signal-to-background spectra are significantly different from those required by higher signal-to-background spectra.

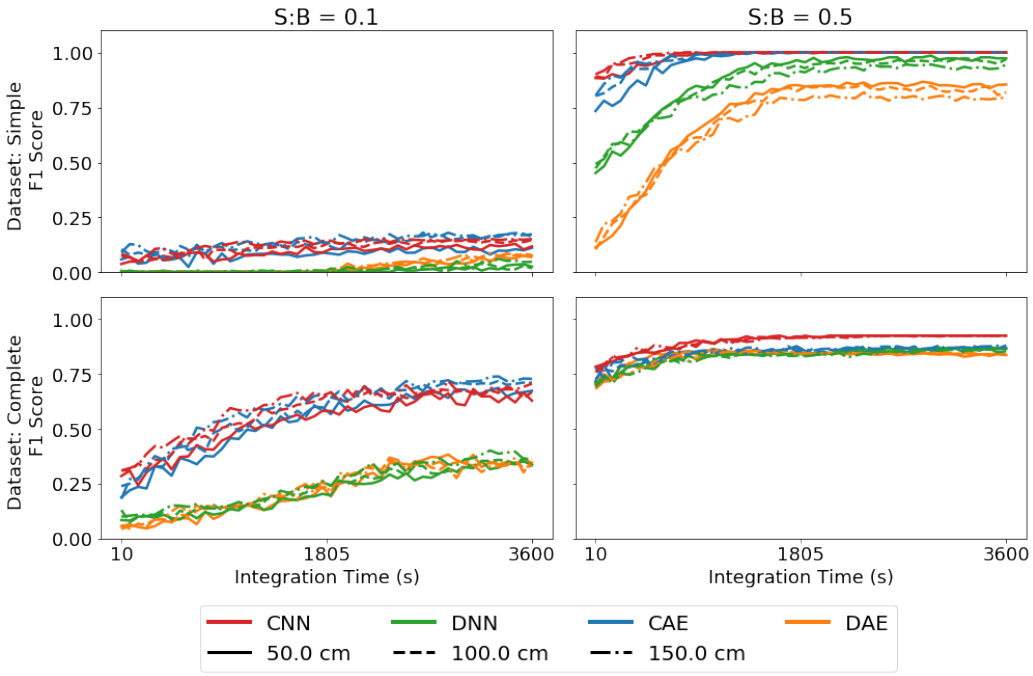


Figure 4.5: F1 score dependence on source-detector height measured using simulated data.

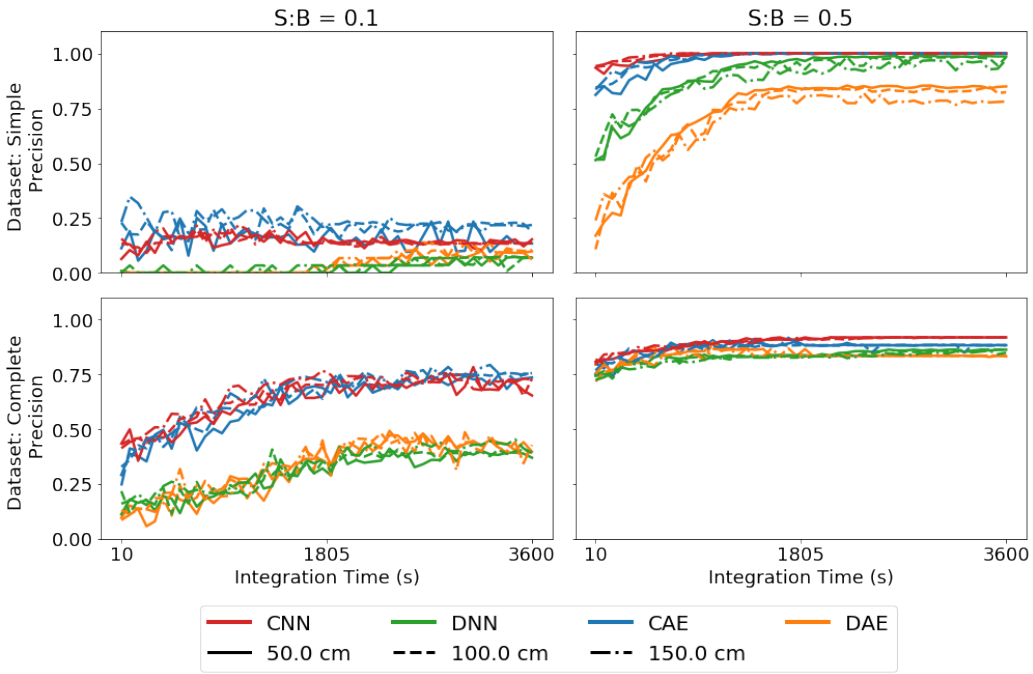


Figure 4.6: Precision dependence on source-detector height measured using simulated data.

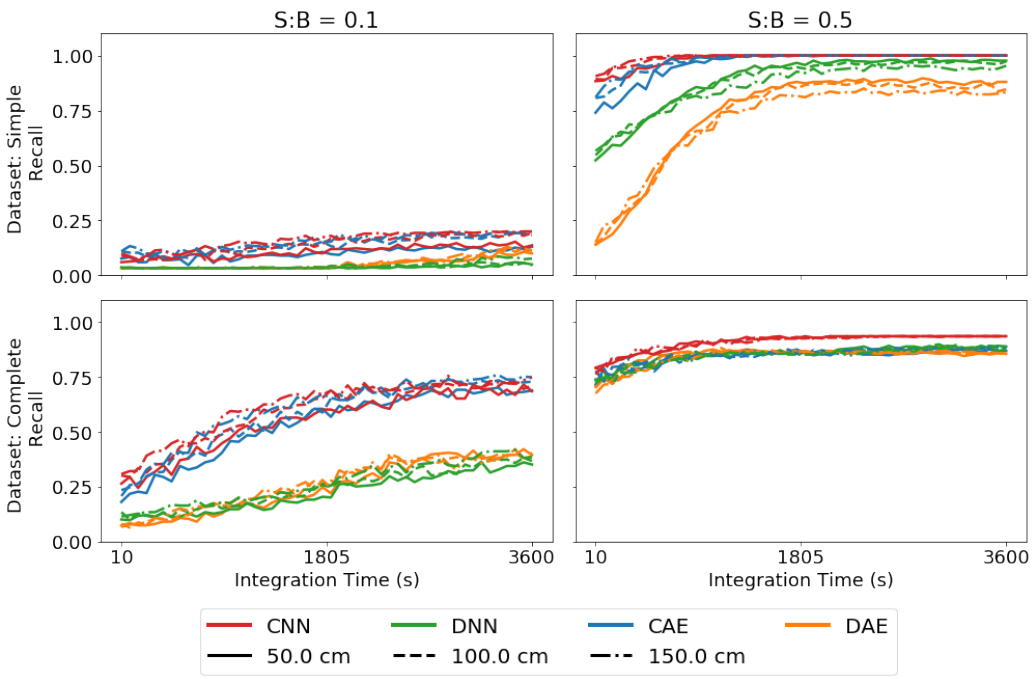


Figure 4.7: Recall dependence on source-detector height measured using simulated data.

4.2.2 Generalization Performance Dependence on Source-Detector Distance

In this section we analyze how each model's F1 score depends on changes in source-detector distance. This tests if each model is sensitive to larger changes in the Compton continuum compared to those associated with changes in source-detector height. An example of these changes was shown previously in Figure 3.3.

As seen in Figures 4.8, 4.9, and 4.10, we observe similar trends seen in the source-detector height generalization such as superior performance of the convolutional models and the poor simple DAE encoding. Changing the source-detector distance changes performance more noticeably compared to changes in standoff distance.

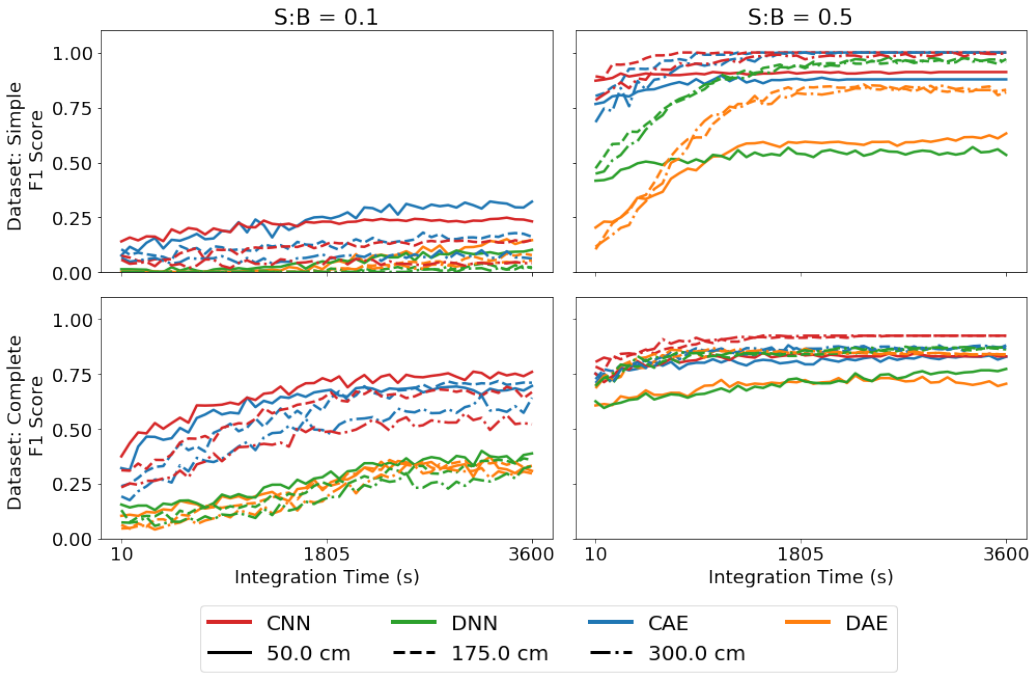


Figure 4.8: F1 score dependence on source-detector distance measured using simulated data.

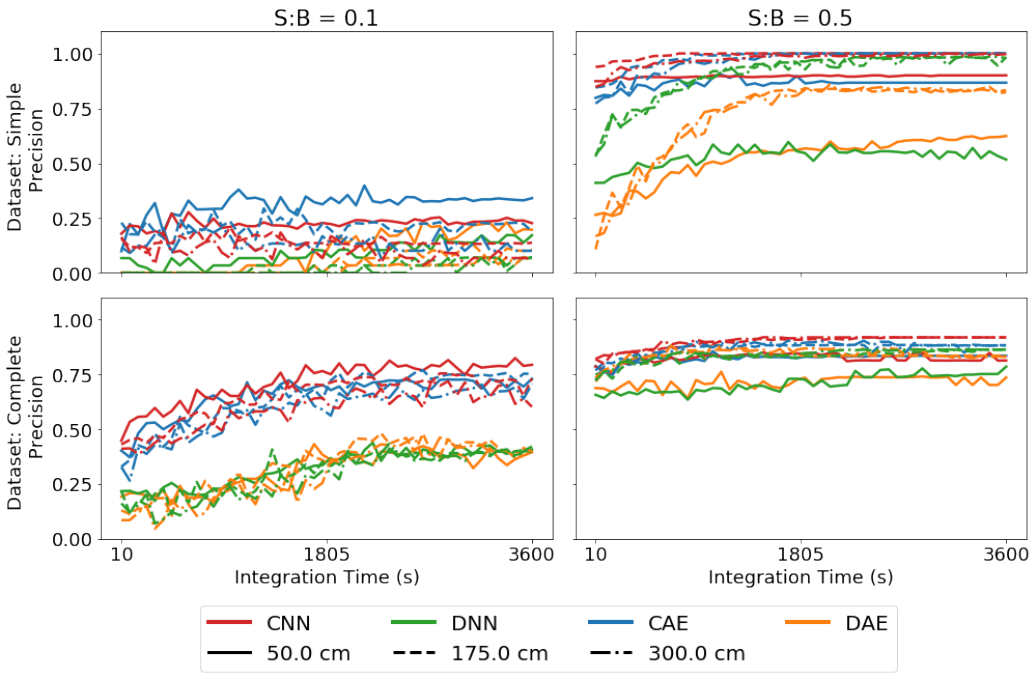


Figure 4.9: Precision dependence on source-detector distance measured using simulated data.

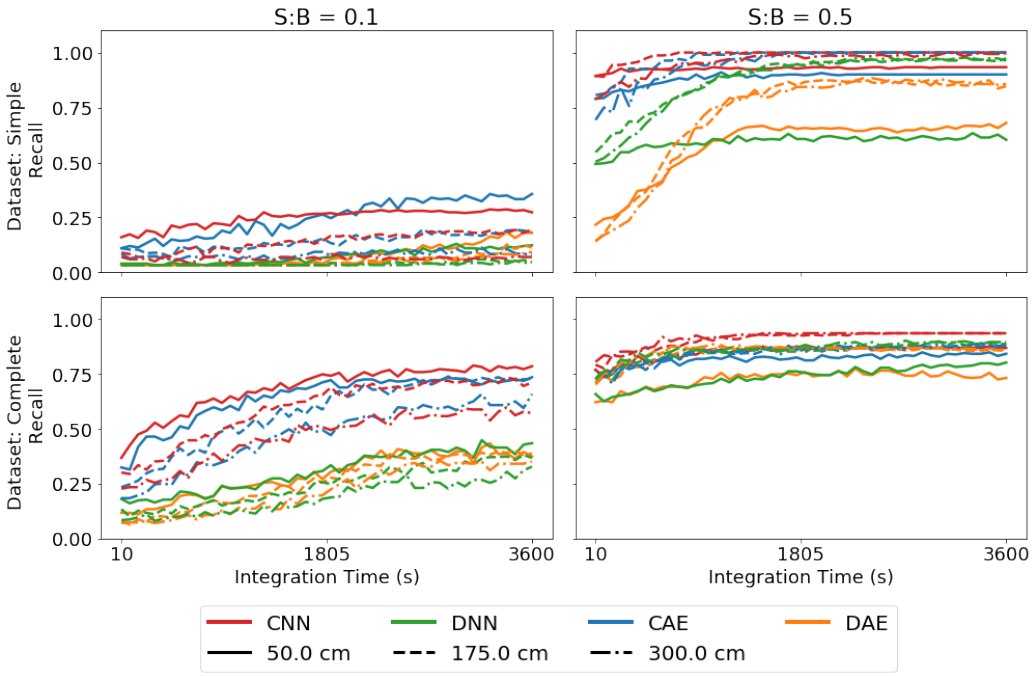


Figure 4.10: Recall dependence on source-detector distance measured using simulated data.

This change is especially noticeable for the shorter standoff distance of 50 cm. At this distance, the peak-to-total ratio increases significantly. The rise in peak-to-total ratio emphasizes the peak information while reducing the Compton continuum's contribution. Each model's F1 score is higher when benchmarked against lower signal-to-background spectra due to the increased photopeak information. This trend does not continue in the higher signal-to-background ratio spectra. At the higher ratio, each model's worst F1 score is achieved when benchmarked against spectra simulated with a 50 cm distance. As seen in Figure 3.3, spectra simulated at a distance of 50 cm have a much different shape than spectra simulated at 175 cm or 300 cm. The simple models only trained with source-detector distances of 175 cm, making identifications for 50 cm spectra very difficult. The complete models trained using each simulated distance. The models could achieve a lower cost function error if they selectively updated weights associated with the 175 cm and 300 cm spectra because they were similar. This explains each complete model's reduced performance when identifying spectra simulated at 50 cm.

This result also shows that each model is sensitive to a spectrum's photopeaks and shape of it's continuum. If the models were only sensitive to

photopeaks, spectra simulated at a distance of 50 cm would have performed best in all cases due to the increased peak-to-total ratio.

4.2.3 Generalization Dependence on Resolution

In this section we analyze how each model's F1 score depends on changes in detector resolution. An example of these changes can be seen in Figures 4.11, 4.12, and 4.13. Changes in resolution do not significantly impact performance of the models. General trends are similar to distance and height generalization.

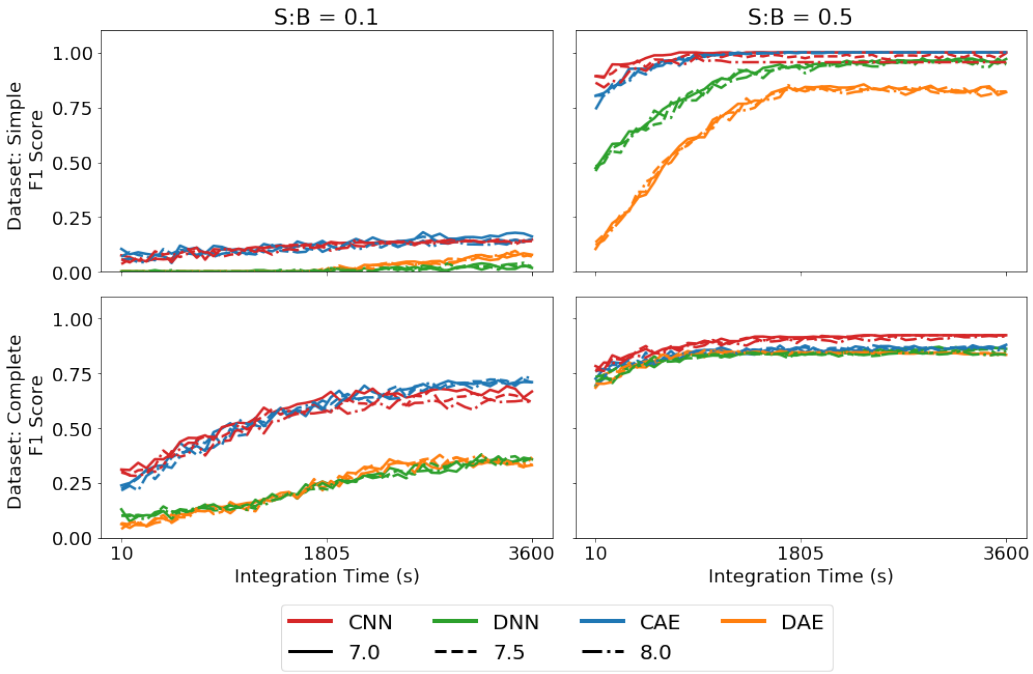


Figure 4.11: F1 score dependence on detector resolution measured using simulated data.

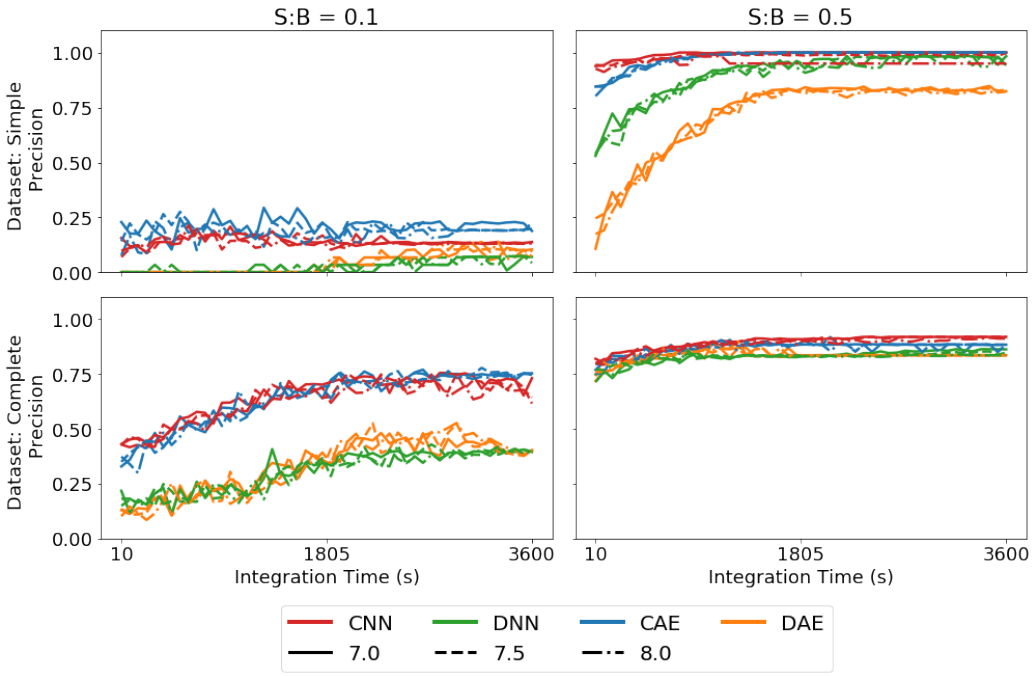


Figure 4.12: Precision dependence on detector resolution measured using simulated data.

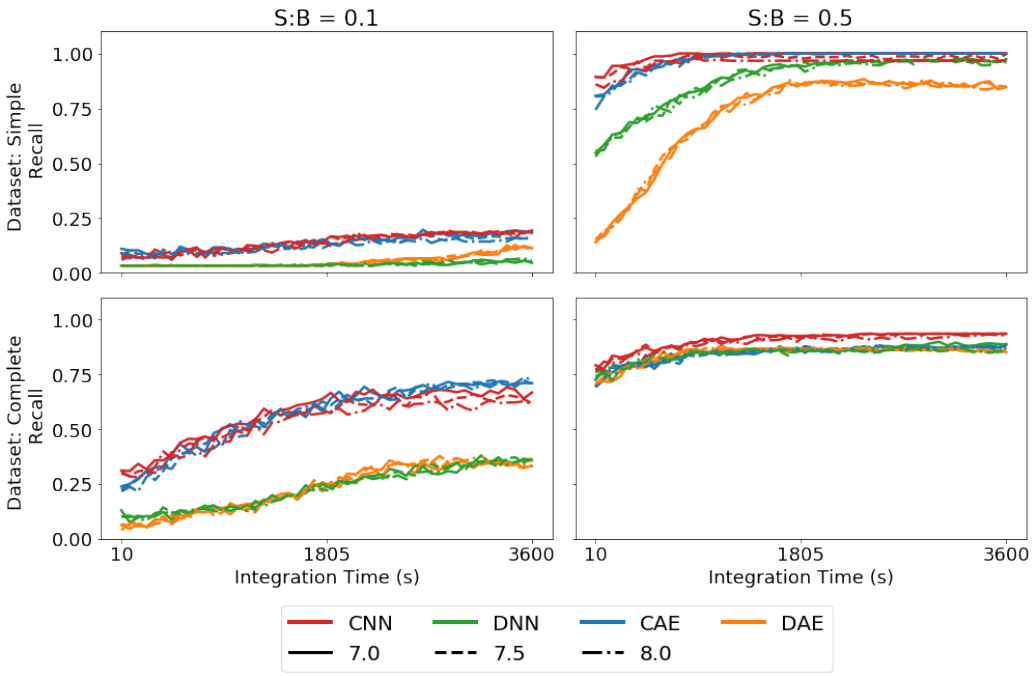


Figure 4.13: Recall dependence on detector resolution measured using simulated data.

4.2.4 Generalization Dependence on Shielding.

In this section we analyze how each model's F1 score depends on changes in shielding. These results are shown in Figures 4.14, 4.15, and 4.16. Spectra were simulated with iron thicknesses that shield 40%, 60%, and 80% of the intensity of a 200 keV gamma-ray.

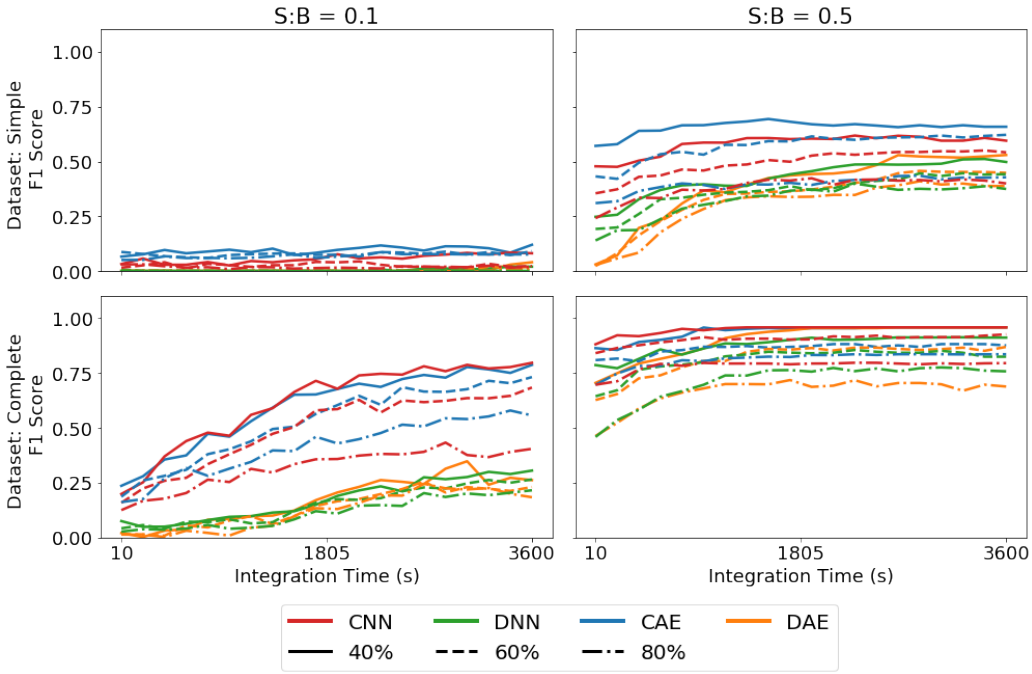


Figure 4.14: F1 score dependence on shielding measured using simulated data.

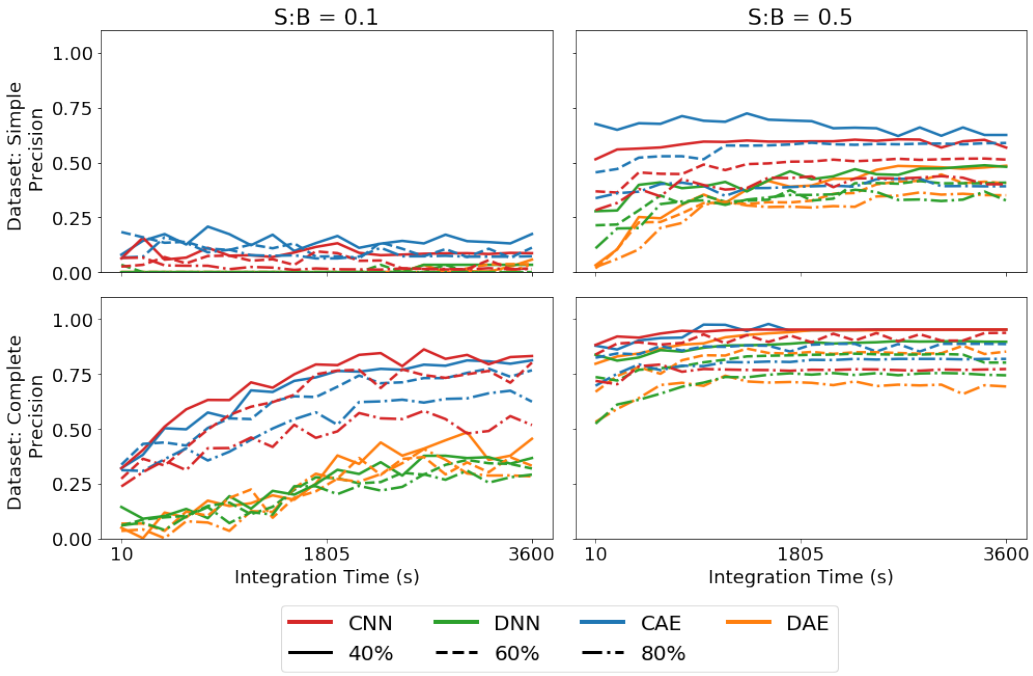


Figure 4.15: Precision dependence on shielding measured using simulated data.

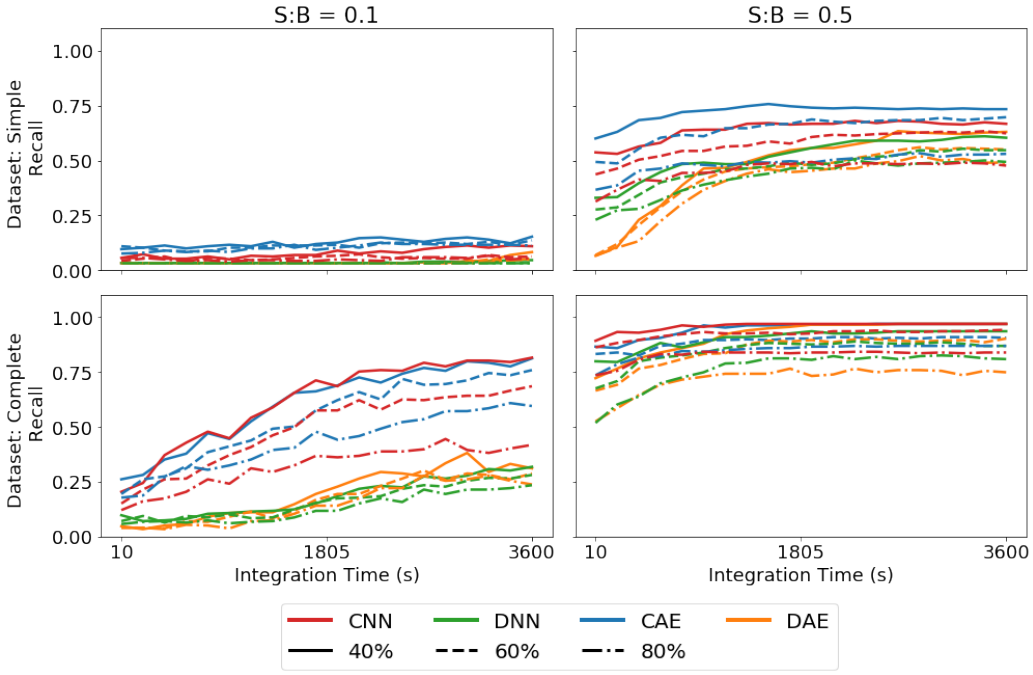


Figure 4.16: Recall dependence on shielding measured using simulated data.

As seen previously, simple models do not generalize to the lower signal-to-background spectra. Simple models, which only use unshielded and 20%

shielded templates, achieve F1 scores between 50% and 75% in higher signal-to-background spectra. This shows that simple models generalize to other shielding thicknesses in higher signal-to-background spectra. The simple CAE outperforms the CNN in spectra with each shielding thicknesses. This demonstrates that the CAE pretraining is useful for conditioning the convolutional weights for identification tasks.

The complete models perform better in the low signal-to-background spectra. Both complete convolutional models perform comparatively and outperform the complete dense models. At 60% and 80% shielding the complete CAE outperformed the CNN in low signal-to-background spectra. This again shows the CAE pretraining boosting the CNN architecture's performance.

4.2.5 Generalization Dependence on Gain

In this section we analyze how each model's F1 score depends on changes in calibration. Spectra were simulated with relative calibration gains of 0.8, 1.0, and 1.2. These relative gain calibrations represent the range of calibrations expected due to temperatures expected when performing source interdiction. The 1.0 gain setting calibrates the 1024^{th} channel of the spectrum to 3 MeV.

As shown in Figures 4.17, 4.18, and 4.19, at the higher signal-to-background ratio, the simple convolutional models achieve F1 scores above 50% for each gain setting except 0.8. Simple dense models achieve F1 scores between 30% and 50% for the same settings.

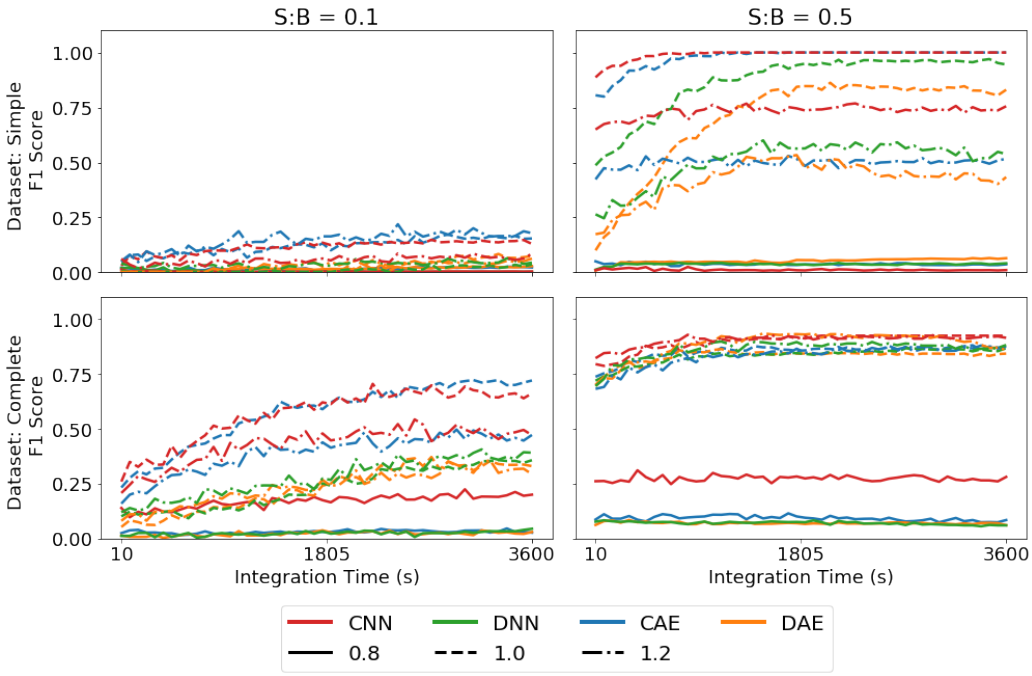


Figure 4.17: F1 score dependence on gain using simulated data.

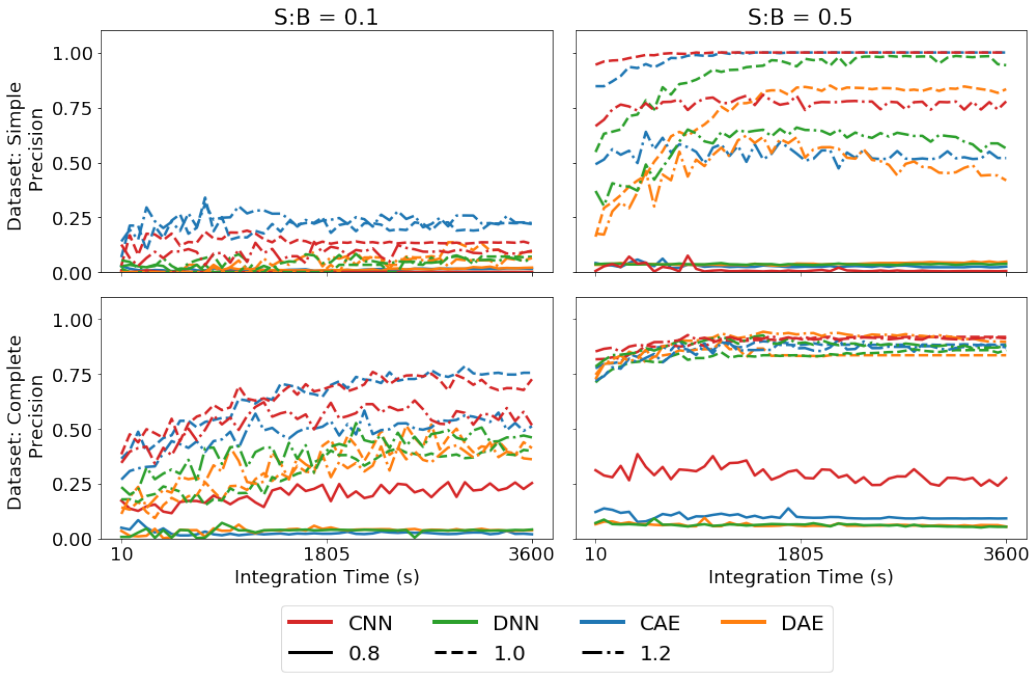


Figure 4.18: Precision dependence on gain using simulated data.

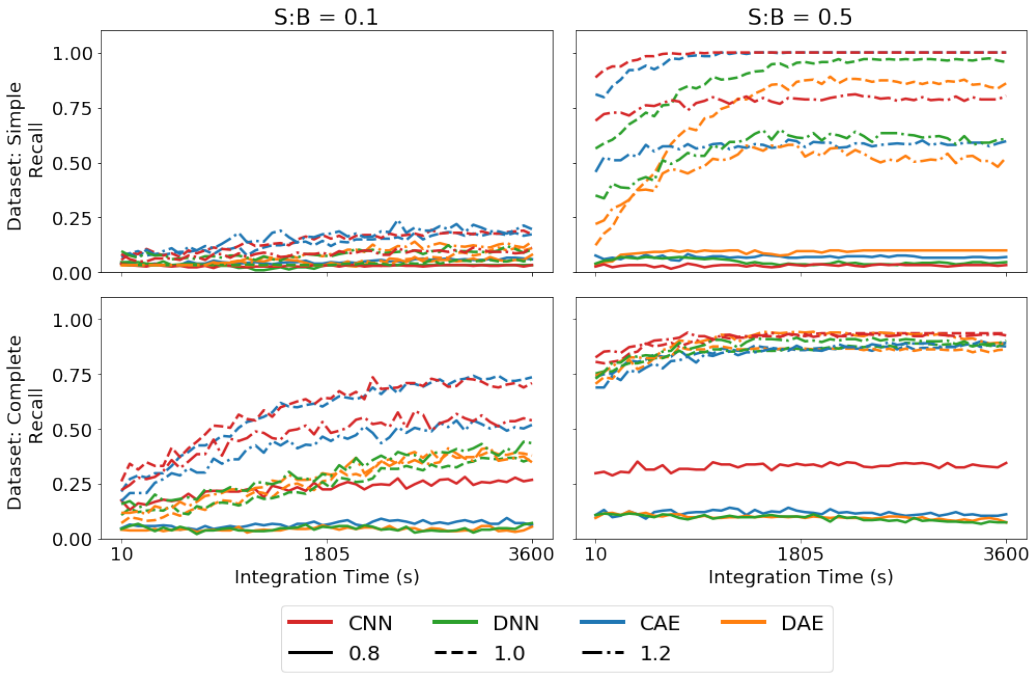


Figure 4.19: Recall dependence on gain using simulated data.

Simple models begin generalizing to spectra with a relative gain of 1.2 while performing poorly on spectra with a relative gain of 0.8. The lower relative gain has the potential to push spectral features into the LLD, removing them from the input signal. Because the simple dataset parameters never obscure these features, the simple models cannot correctly identify spectra without them.

The complete models performed better with changing gain, but still struggled with the 0.8 relative gain for the reasons outline in the previous paragraph. Compared to other complete models, the complete CNN performed the best on each relative gain setting.

4.3 Performance Identifying Measured Spectra

In this section we investigate each ANN's performance when identifying real gamma-ray spectra with calibration settings and shielding possible in source interdiction activities. To measure performance, we observe how each model's posterior probability for each isotope changes over integration times ranging from 10s to 10 mins. Sources included are ^{137}Cs, ^{60}Co, ^{133}Ba, and ^{152}Eu. ^{137}Cs and ^{60}Co are included because they have uncomplicated spectra, ^{137}Cs with

only one primary photopeak at 662 keV and ^{60}Co with two photopeaks at 1173 and 1332 keV. These are also common industrial and medical sources which are important for source interdiction. ^{133}Ba and ^{152}Eu have more complicated spectra with many photopeaks. Photopeaks from ^{133}Ba are all below 400 keV and the photopeaks from ^{152}Eu range from 121 keV to 1213 keV [90]. ^{133}Ba is included in these tests because it is a medical isotope and an often used surrogate for plutonium. ^{152}Eu is included because it's large range of photopeak energies are uniquely effected by shielding and calibration changes. Shielding ^{152}Eu can remove low-energy photopeaks while only slightly reducing higher-energy photopeaks. This presents a unique challenge for identification algorithms.

A diagram of the laboratory setup used for these measurements is shown in Figure 4.20. The radioactive sources, 1 inch diameter by 1/8 inch thick plastic disks containing μCi quantities of radioactive material, were measured on a wooden desk approximately one meter from the nearest wall to reduce environmental scatter. We used a 2x2 inch Ortec NaI detector to measure the sources. The source-detector distance, measured from the source disk's face to the detector's face, was adjusted to keep the signal-to-background ratio either 0.5 or 1.0. Blocks of shielding material were added for the measurements in Section 4.3.2. Measurements in Section 4.3.1 used no shielding material.

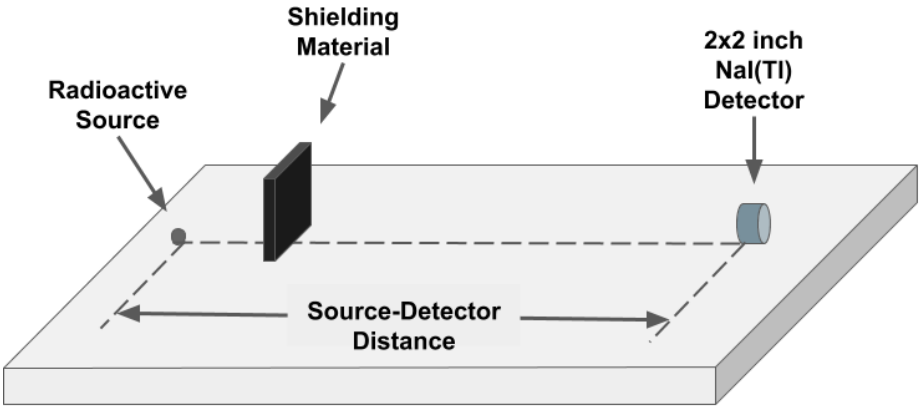


Figure 4.20: Diagram of the laboratory setup used to measure radioactive sources.

For this section, we define a model's performance to be good when the posterior probability increases above 50 % after 60 seconds. This measurement

time is typical for source interdiction tasks. We also trigger an alarm when an isotopes posterior probability rises above 50%. Plots in this section observe true positives, when the measured isotope's posterior probability exceed the 50% threshold, and false negatives, when the measured isotope's posterior probability is below the 50% threshold. For practical implementation, this threshold needs to be tuned for specific operational requirements.

4.3.1 Model Performance on Changing Voltage

To test how well each model identified spectra with changing calibration, spectra were measured with different PMT voltages. PMT voltages included 720 V, 745 V, 770 V, and 795 V. The detector calibration of 770 V calibrated the detector's 1024^{th} channel to 3 MeV. This gain setting was used as the reference calibration. Relative gain settings for the PMT voltages are measured with respect to the relative shift of the 662 keV photopeak. The source-detector distance of each source was adjusted so the ratio of counts per second from the source and background were 0.5 and 1.0. Example spectra measured at the experiment's time limits are shown for each isotope.

4.3.1.1 Identification of ^{137}Cs

Example ^{137}Cs spectra identified in this analysis are shown in Figure 4.21. The 662 keV photopeak is clearly visible even at a measurement time of 10 s. At a 10s integration time, background peaks from ^{40}K, at 1460 keV, and the thorium series, 2614 keV, are not visible. Without these peaks, it is very difficult to know the energy calibration of the detector.

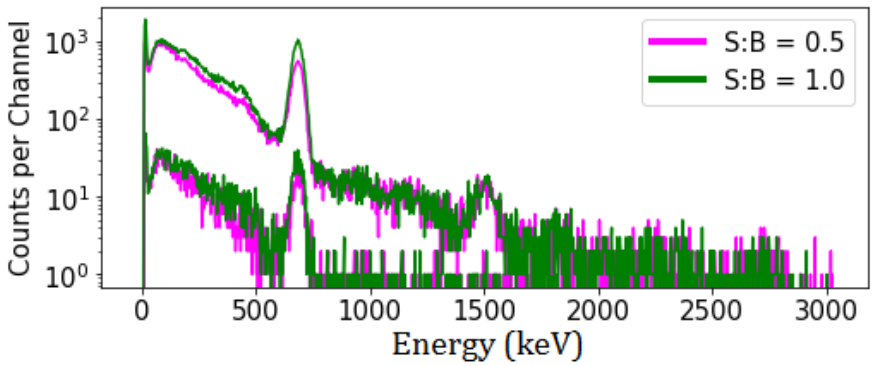


Figure 4.21: Example ^{137}Cs measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively.

Figure 4.22 demonstrates the effect of measurement time on each model's posterior probability for ^{137}Cs. As time increases, the simple DNN and DAE monotonically increase their probabilities for ^{137}Cs measured with the default gain. These probabilities approach an asymptote in the higher signal-to-background spectra. The simple CNN jumps to 100% posterior probability for default gain spectra. This very high confidence from the CNN is likely due to fact that the single photopeak spectrum is easy to identify. Complete models demonstrate more monotonic performance for a wider range of gain settings. Monotonic increases in posterior probability are desirable for isotope identification algorithms because they are more easily interpreted by an unskilled user.

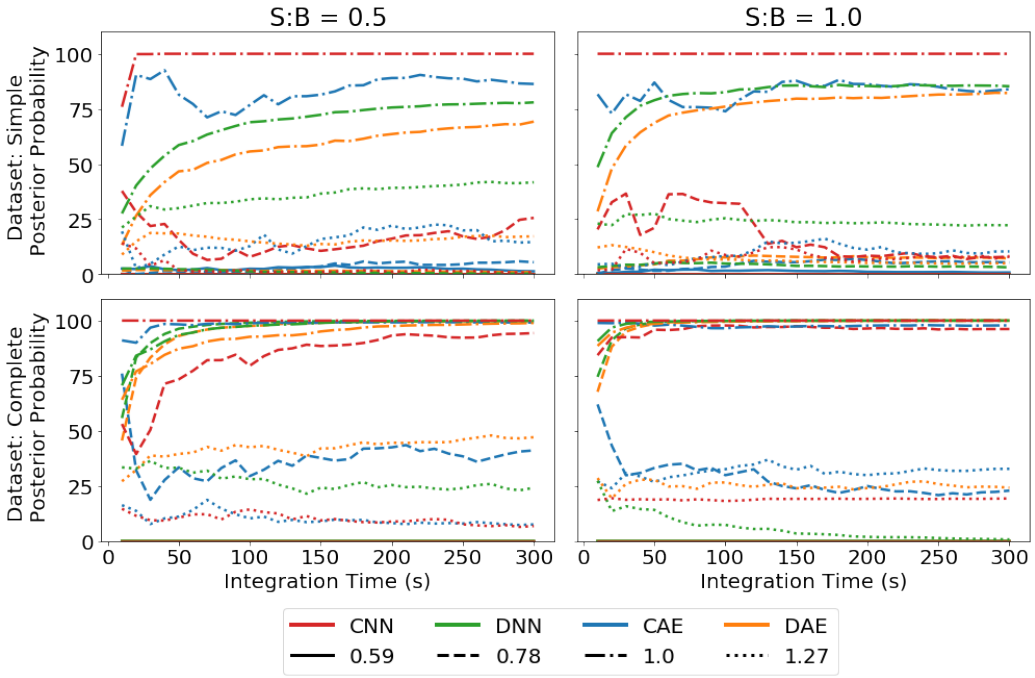


Figure 4.22: Effect of calibration gain on the posterior probability of ^{137}Cs measured using real ^{137}Cs spectra.

Each ANN, except the simple DAE, identifies reference calibration spectra above the 50% posterior probability threshold after 60s of measurement. Simple ANNs fail to alarm on ^{137}Cs spectra with other calibration settings. Complete ANNs, with the exception of the CAE, successfully alarm when identifying spectra with a relative gain of 1.0 and 0.78.

Spectra with relative gains of 0.59 and 1.27 never raise a true positive alarm. These settings are too far outside of the training dataset's relative gain ranges, (0.8, 1.2), for correct identification. Correctly identifying ^{137}Cs with a relative calibration of 1.27 are possible if the detection threshold is lowered, but this risks increasing the false positive rate. Spectra with a relative calibration of 0.59 move the 662 keV photopeak to 340 keV. As shown in Table 4.4, ANNs mistakes ^{137}Cs at this calibration for other isotopes with lower-energy photopeaks. The simple CNN raises an incorrect alarm on ^{192}Ir, which has it's largest photopeak at 316 keV. The complete CNN, DNN, and DAE also raise incorrect alarms. Both complete convolutional models predict the same isotope, ^{131}I, which has a primary photopeak at 364 keV. Both dense models predict ^{239}Pu.

Table 4.4: ANN predictions of ^{137}Cs spectra measured with a relative gain shift of 0.59. Spectra were measured with a signal-to-background ratio of 1.0 for 60 s. Simple and complete ANNs are indicated by the S- and C- prefix respectively. Incorrect alarms are bolded.

	S-CNN	**S-DNN**	**S-CAE**	**S-DAE**
Prediction	**^{192}Ir**	^{177m}Lu	^{75}Se	^{67}Ga
Probability	**80**	33	41	33

	C-CNN	**C-DNN**	**C-CAE**	**C-DAE**
Prediction	**^{131}I**	**^{239}Pu**	^{131}I	**^{239}Pu**
Probability	**100**	**70**	40	**61**

Even for isotopes with a single photopeak, including a wide range of calibrations in a ANNs training dataset is necessary for good performance when identifying measured spectra.

4.3.1.2 Identification of ^{60}Co

As seen in Figure 4.23, the 1173 keV and 1332 keV photopeaks from ^{60}Co spectra measured at 10s each have a maximum number of counts less than 100. This is around a factor of two less than the maximum photopeak counts in the ^{137}Cs spectra shown in the previous section. This amplifies the effects of Poisson noise on the ^{60}Co spectra when compared to the ^{137}Cs spectra.

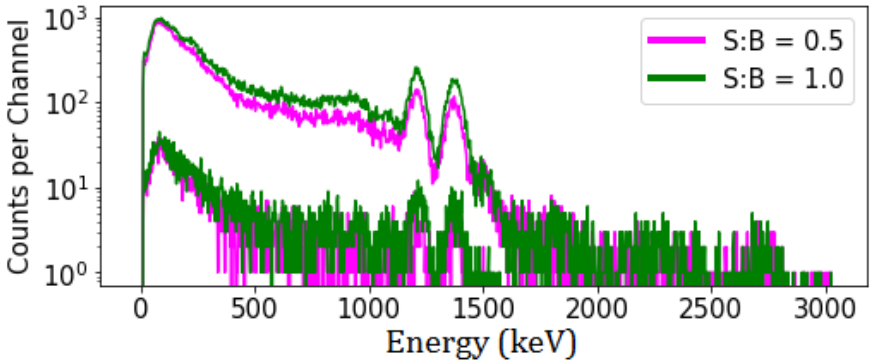


Figure 4.23: Example ^{60}Co spectra measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively.

Figure 4.24 shows that monotonic behavior is observed in more ^{60}Co identifications in a wider range of calibrations than observed for ^{137}Cs. Similar to identifications for ^{137}Cs, shown in Figure 4.22, the convolutional models exhibited more erratic behavior than dense models.

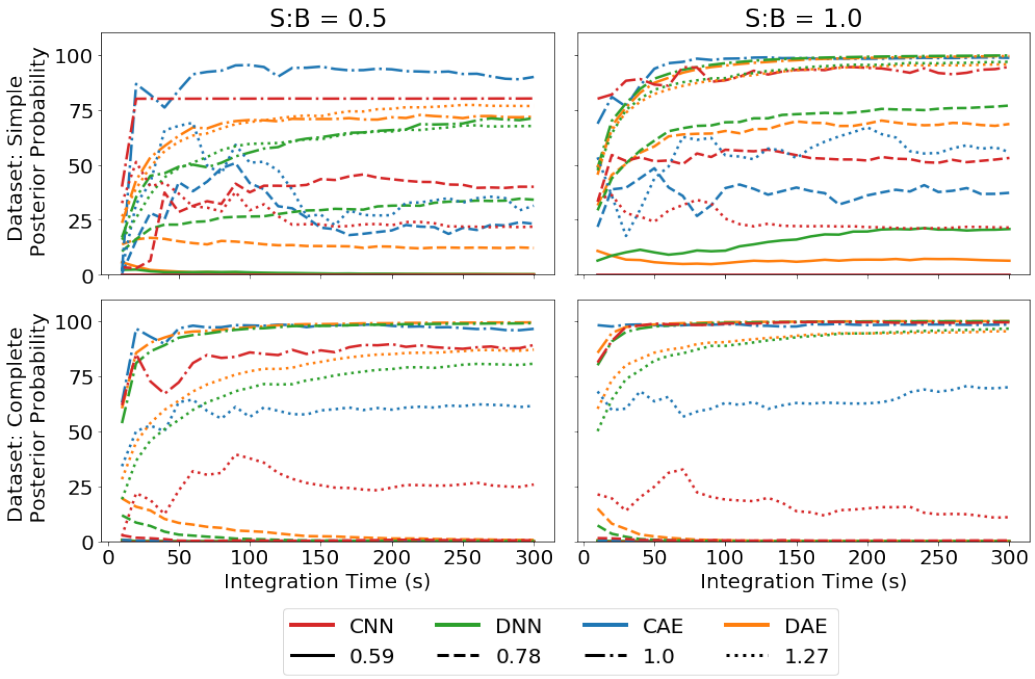


Figure 4.24: Effect of calibration gain on the posterior probability of ^{60}Co measured using real ^{60}Co spectra.

Simple ANNs correctly alarm on ^{60}Co spectra measured using the default gain. This effect is also seen in the ^{137}Cs identifications. Unlike identifications in ^{137}Cs, both simple dense ANNs correctly alarm on the gain setting of 1.27. This shows that dense networks are more flexible when identifying spectra with parameters outside of their training set. Convolutional models may develop feature extraction mechanisms that are more prone to overtraining than dense models. Simple dense ANNs outperforming simple convolutional ANNs is also shown when identifying spectra measured with the higher signal-to-background ratio. Simple dense ANNs correctly alarm on spectra with all gain settings except 0.59.

Complete ANNs fail to alarm on spectra with a relative gain setting of 0.78. ANNs predict these spectra to be ^{238}U with posterior probabilities above 98%. The ^{238}U spectrum is characterized by photopeaks at 93 keV, 766 keV, and 1001 keV. With sufficient shielding, the 93 keV photopeak is

completely attenuated, leaving a two photopeaks. These photopeaks are have an energy similar to the photopeaks from ^{60}Co. By including shielded ^{238}U spectra in the complete training dataset, we introduced spectra similar to ^{60}Co.

4.3.1.3 Identification of ^{133}Ba

^{133}Ba spectra are shown in Figure 4.25. Spectra measured for 10s have features and photopeaks difficult to identify by eye. Due to a measurement error, ^{133}Ba spectra measured at a signal-to-background ratio of 0.5 with a relative gain of 1.27 were lost. Spectra measured with this setting are replaced by spectra measured with a signal-to-background ratio of 0.5 and relative gain of 1.0 recalibrated to match a gain of 1.27.

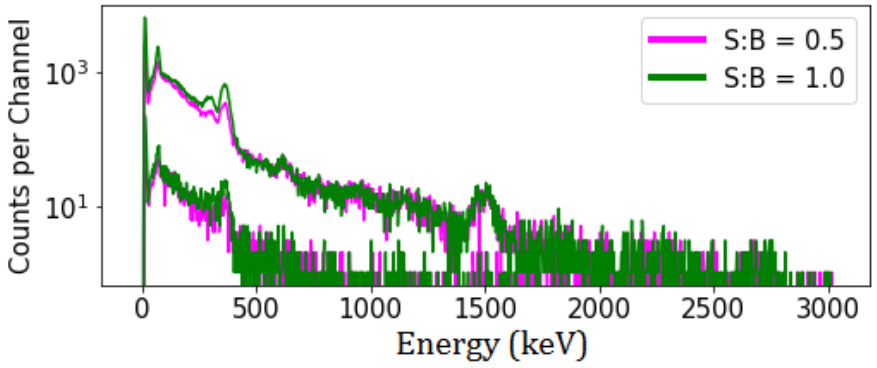


Figure 4.25: Example ^{133}Ba spectra measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively.

Identification performance is shown in Figure 4.26. In most cases, dense models fail to achieve posterior probabilities above 20% when identifying ^{133}Ba spectra. Only the simple CAE with a source-to-background ratio of 1.0 alarms after 120s measurement time. The simple CAE only correctly identifies spectra measured using the default calibration. These results show that simple ANNs will not identify ^{133}Ba in spectra measured using calibrations outside the simple training dataset's parameters.

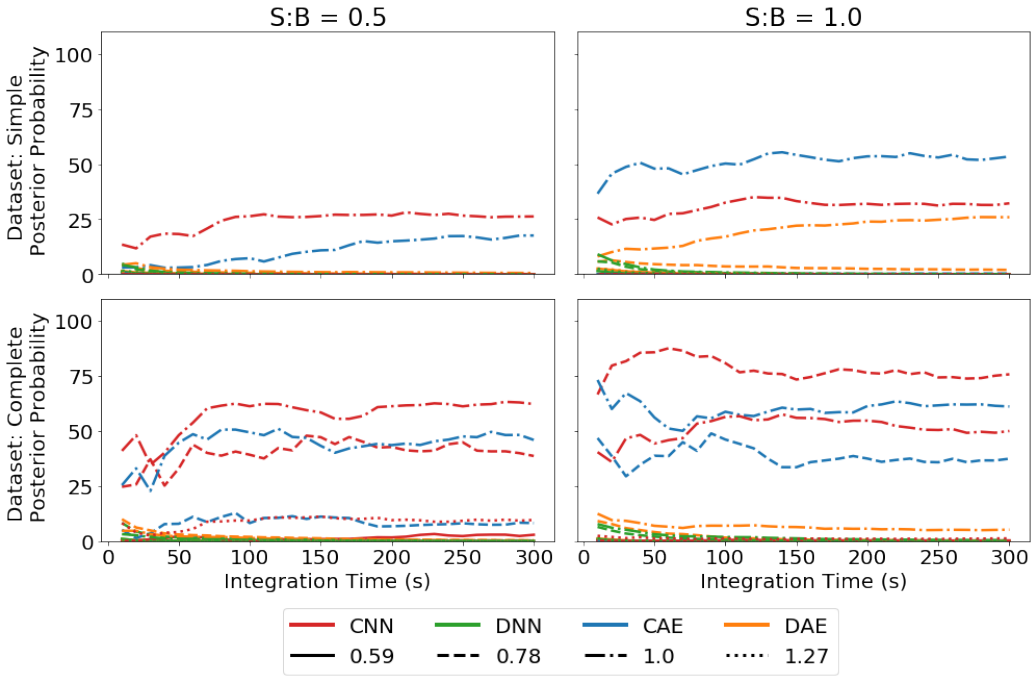


Figure 4.26: Effect of calibration gain on the posterior probability of ^{133}Ba measured using real ^{133}Ba spectra.

Only the complete CNN successfully alarms on ^{133}Ba with a signal-to-background ratio of 0.5. Complete CNN identification performance for spectra measured with gain settings of 1.0 and 0.78 improve at the higher signal-to-noise ratio. This shows that despite their erratic predictions, convolutional models may be necessary to identify low-energy photopeak spectra similar to ^{133}Ba. Adding relative gain settings using a finer sampling on the range (0.8, 1.2) may be necessary to consistently identify ^{133}Ba spectra.

4.3.1.4 Identification of ^{152}Eu

Spectra of ^{152}Eu are shown in Figure 4.27. ^{152}Eu emits photons with a wide range of energies including: 122 keV, 344 keV, 779 keV, 963 keV, 1086 keV, 1112 keV, and 1408 keV.

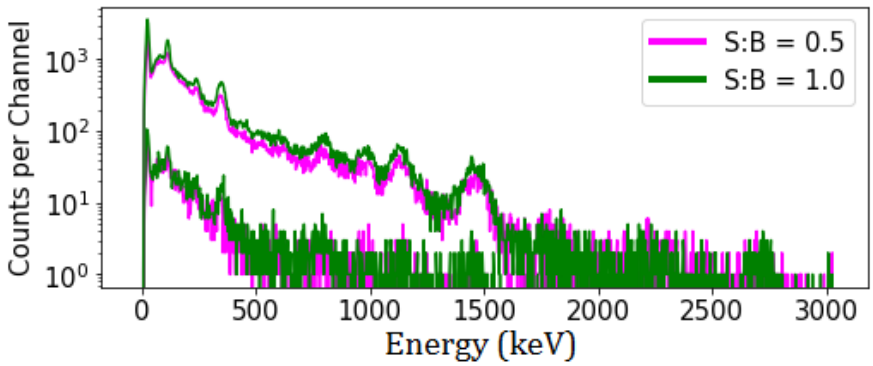


Figure 4.27: Example ^{152}Eu spectra measured with signal-to-background ratios of 0.5 and 1.0. The lower and higher count spectra use a measurement times of 10s and 300s respectively.

As seen in Figure 4.28, simple models occasionally correctly alarm on only relative gain settings of 0.78 and 1.0. More consistent performance is demonstrated when complete models identify spectra at the higher signal-to-background ratio. Even in this setting, relative gain settings of 0.59 and 1.27 are not identified correctly. This shows that ANNs cannot correctly identify spectra with relative gain settings far outside the range used by the training dataset.

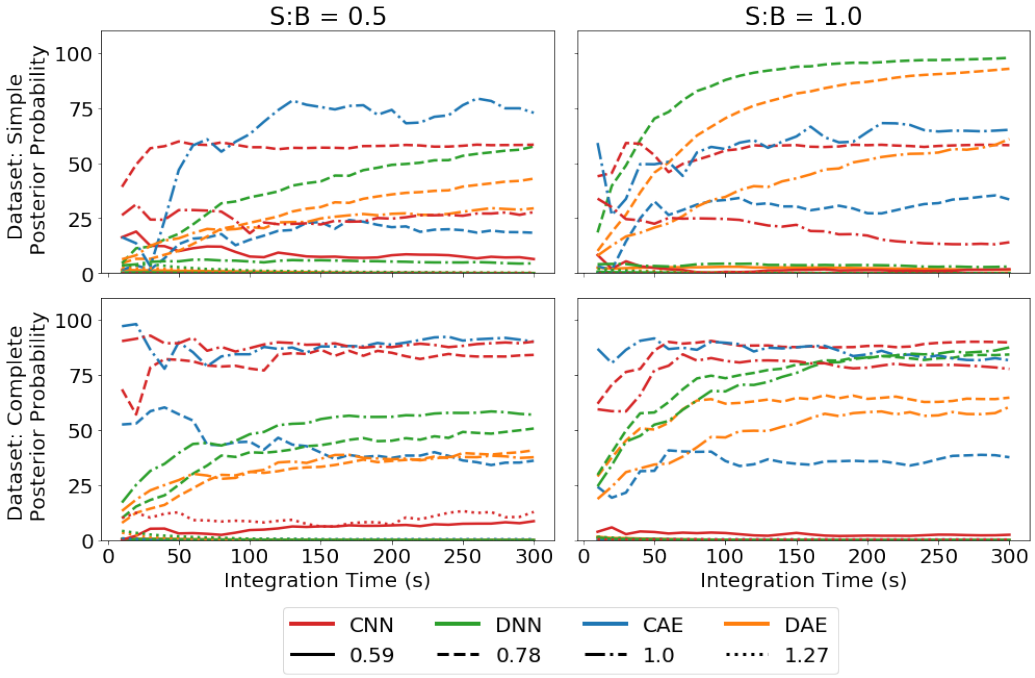


Figure 4.28: Effect of calibration gain on the posterior probability of ^{152}Eu measured using real ^{152}Eu spectra.

4.3.2 Model Performance with Respect to Shielding

To investigate the identification performance of each ANN with increased shielding thicknesses, sources were shielded with increasingly thick iron or aluminum blocks and measured. For each measurement, the source-detector distance was adjusted so the ratio of counts per second measured by the detector was equal to the counts per second from background. Each spectrum was measured using the default detector PMT calibration of 770 V.

4.3.2.1 Identification of Shielded ^{137}Cs

Figure 4.29 shows that shielding affects the ^{137}Cs spectrum by reducing the 662 keV photopeak's peak-to-total ratio. Shielding does not visibly impact the shape of Compton continuum.

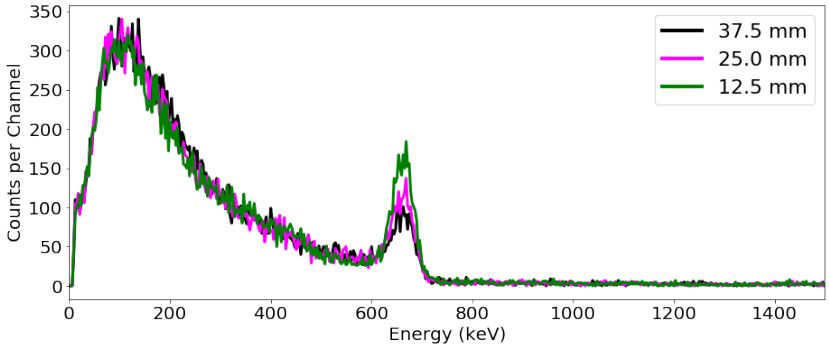


Figure 4.29: Spectra of ^{137}Cs shielded with increasing amounts of iron measured for 60 s.

As seen in Figure 4.30, all simple and complete models correctly alarm on each shielded ^{137}Cs isotope. Shielding impacts the simple dense models' performance more than the convolutional models. This reduction in performance is demonstrated by a reduced posterior probability. The DAE outperforms the DNN, achieving a higher posterior probability for each shielding thickness after 50 s. This demonstrates that the dense pretraining improved the dense model's performance.

Each complete model achieves a posterior probability above 90% for all measured times and shielding thicknesses. This shows that the additional parameters in the complete training dataset boost performance in trained ANNs when identifying shielded isotopes.

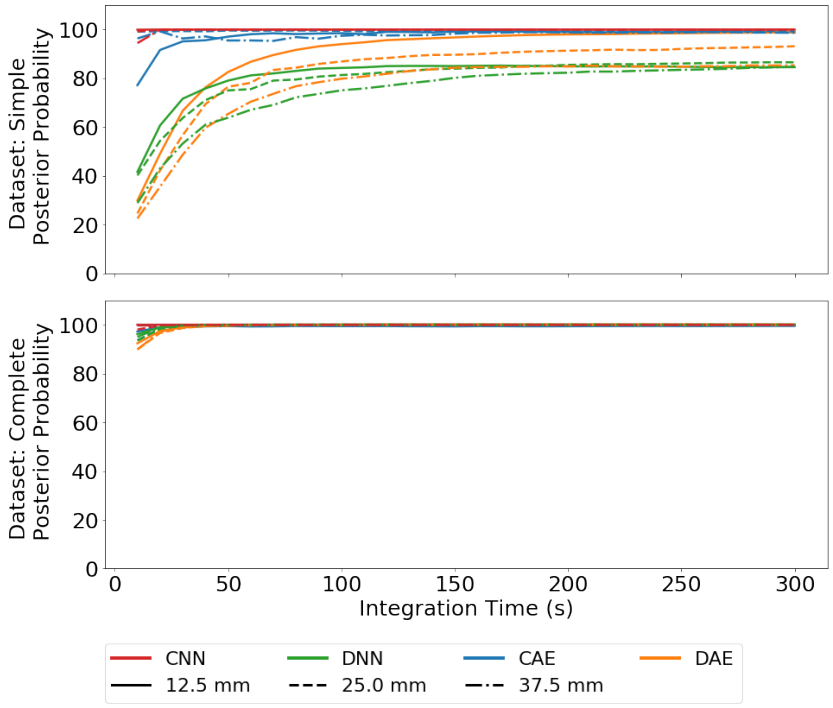


Figure 4.30: Effect of iron shielding thickness on the posterior probability of ^{137}Cs measured using real shielded ^{137}Cs spectra.

4.3.2.2 Identification of Shielded ^{60}Co

As shown in Figure 4.31, the effect of shielding is not visibly significant in ^{60}Co spectra. The 1173 keV and 1332 keV photopeaks from ^{60}Co are attenuated less than the 662 keV photopeak from ^{137}Cs. Shielding only slightly decreases the area of the ^{60}Co photopeaks.

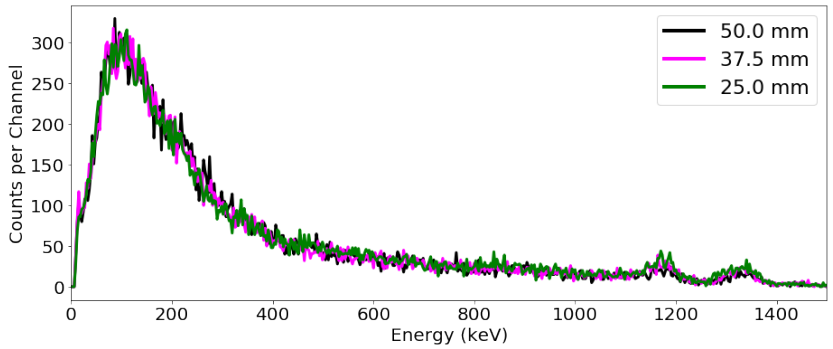


Figure 4.31: Spectra of ^{60}Co shielded with increasing amounts of iron measured for 60 s.

As seen in Figure 4.32, each simple convolutional model successfully alarms

on each shielded ^{60}Co spectrum. In particular, the CAE identifies each shielded ^{60}Co spectrum with a near 100% posterior probability after 20 s. This demonstrates that the natural feature extraction capabilities of convolutional models work well when identifying simple spectra with shielding amounts not included in their training dataset. This also shows that using autoencoder pretraining increases the convolutional model's performance.

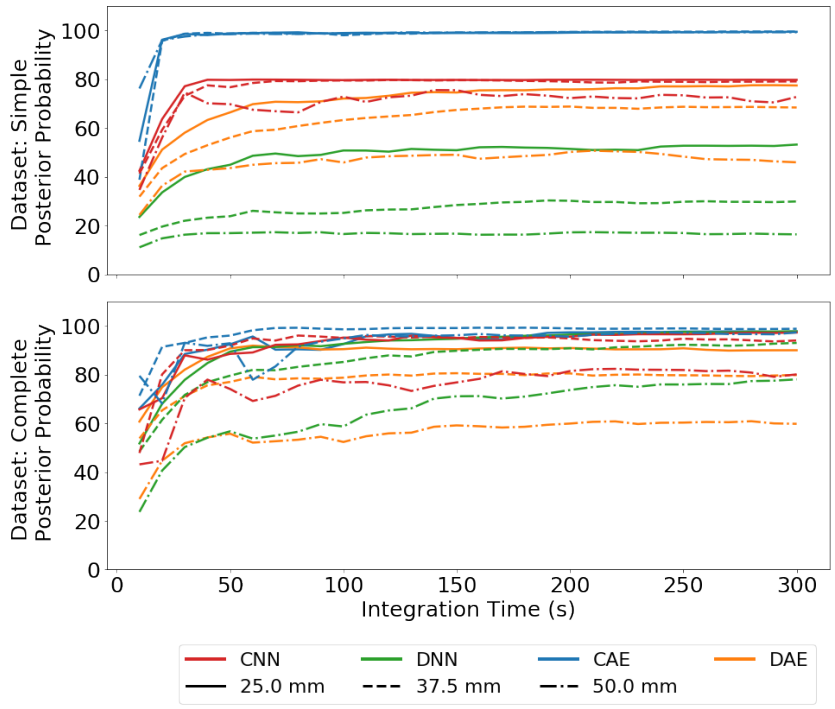


Figure 4.32: Effect of iron shielding thickness on the posterior probability of ^{60}Co measured using real shielded ^{60}Co spectra.

Shielding has a larger detrimental impact on simple dense ANNs compared to simple convolutional ANNs. Despite a clear photopeak at 662 keV, only the DAE successfully alarms on ^{60}Co shielded with the two thinnest iron blocks. As with the convolutional model, this result shows that dense models visibly improve with autoencoder pretraining. The poor performance of the simple dense models is attributed to overtraining on the single ^{60}Co example in the training dataset.

Each model's performance improves when trained using the complete dataset. All complete ANNs successfully alarm on spectra measured with each shielding thickness. Both complete convolutional ANNs have comparable performance. Similar to the simple models, complete dense ANNs achieve posterior

probabilities less than the comple convolutional ANNs.

4.3.2.3 Identification of Shielded ^{133}Ba

Due to the relatively low gamma-ray energies of ^{133}Ba compared to the other spectra in this section, aluminum with thicknesses of 6.56 mm, 25.5 mm, and 51.0 mm were used as the shielding material. Characteristic ^{133}Ba photons have relatively low energy - particularly the photopeaks at 31 and 81 keV. As seen in Figure 4.33, shielding significantly distorts the spectrum at these energies.

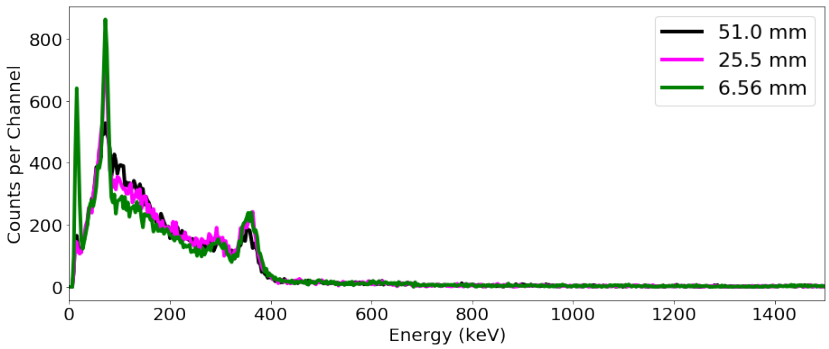


Figure 4.33: Effect of aluminum shielding thickness on the posterior probability of ^{133}Ba measured using real shielded ^{133}Ba spectra.

As seen in Figure 4.34, all simple ANNs, except the simple CAE, fail to correctly alarm when identifying shielded ^{133}Ba spectra. The simple CAE correctly alarms only when identifying ^{133}Ba with the thinnest shielding. This shows that the models trained using the simple dataset will perform poorly when measuring shielded spectra visibly distorted by shielding material.

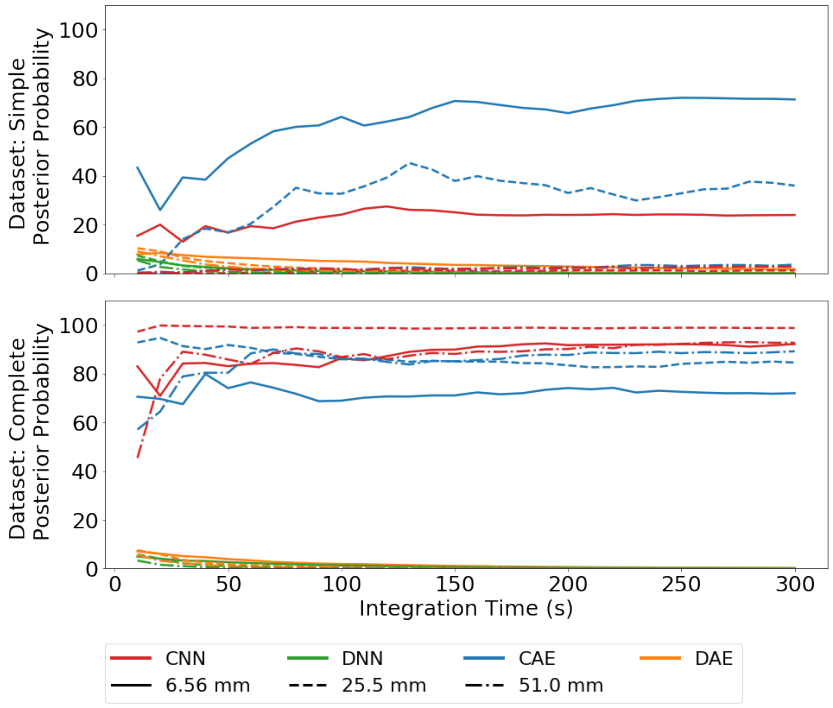


Figure 4.34: Shielding generalization performance in real ^{133}Ba spectra.

Each complete convolutional model performs well on all shielded ^{133}Ba spectra. Each model successfully alarms after 20 s. Dense models perform poorly, incorrectly alarming on ^{131}I in spectra with the thinnest shielding and ^{51}Cr in spectra with thicker shielding material. The largest photopeak from ^{131}I is located at 364 keV and the single photopeak from ^{51}Cr is located at 320 keV. The complete dense ANNs poor performance when identifying shielded ^{133}Ba indicates that they are poor choices for differentiating spectra with low-energy photopeaks. This is concerning because medical sources, plutonium, and uranium all have characteristic photopeaks below 400 keV. These low-energy photopeak sources are particularly important to identify and differentiate for source interdiction.

4.3.2.4 Identification of Shielded ^{152}Eu

As seen in Figure 4.35, ^{152}Eu has many photopeaks spread across a large range of energies. As with ^{133}Ba, higher-energy photopeaks (like those at 779 keV, 963 keV, 1086 keV, 1112 keV, and 1408 keV) are attenuated less than lower-energy photopeaks (like those at 122 keV and 344 keV).

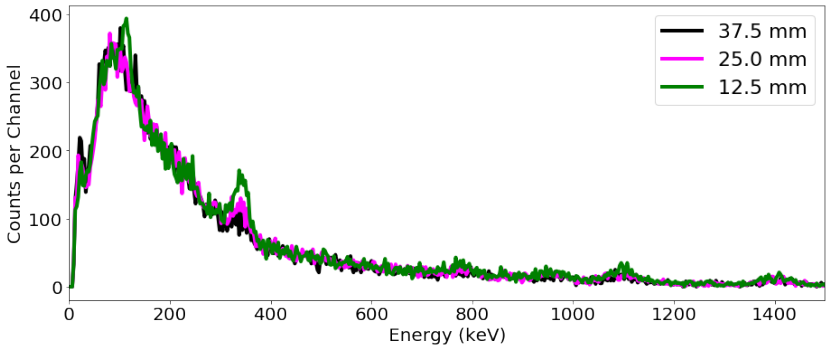


Figure 4.35: Spectra of ^{152}Eu shielded with increasing amounts of iron measured for 60 s.

Identification results for shielded ^{152}Eu are shown in Figure 4.36. Similar to identifications in shielded ^{133}Ba, the simple CAE correctly alarms on ^{152}Eu spectra shielded with the two thinnest shielding materials. In addition to this, the simple DNN correctly alarms on spectra measured with the lightest shielding thickness.

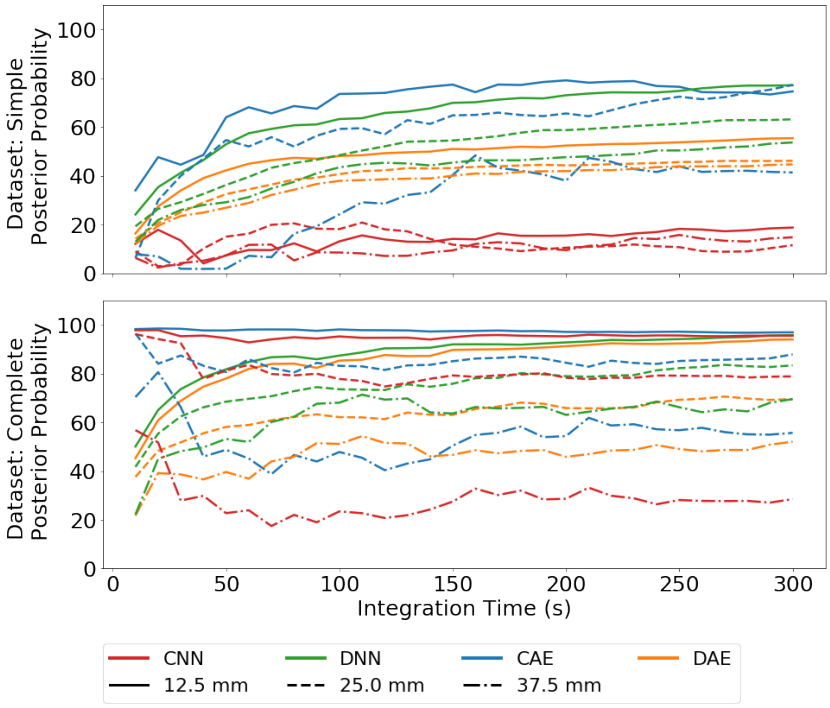


Figure 4.36: Effect of iron shielding thickness on the posterior probability of ^{152}Eu measured using real shielded ^{152}Eu spectra.

All complete models successfully alarm on ^{152}Eu spectra shielded with the two thinnest blocks of iron. Both ^{152}Eu and ^{133}Ba have a large number of

photopeaks compared to ^{60}Co and ^{137}Cs. Despite the large number of photopeaks, ANNs correctly identify ^{152}Eu more often than ^{133}Ba. The difference between these two isotopes is that ^{152}Eu has a unique set of photopeaks relative to the library of training isotopes. The unique set of photopeaks in a wide range of energies make identification easier for the ANNs. As previously discussed, ^{133}Ba possesses a large number of photopeaks in the lower-energy region shared by other isotopes' photopeaks. To improve identification of these important low-energy isotopes, additional examples of them need to be included in each ANN's training dataset.

4.4 Comparison with Peak-Based Bayesian Classifier

In this section we compare our trained ANNs to the performance of a peak-based Bayesian classifier created in Dr. Jacob Stinnett's PhD dissertation [91]. The peak-based classifier extracts photopeak locations and areas using a wavelet transform. These features are then used in a Bayesian classifier to produce posterior probabilities for a library of ANSI N42-34-2006 standard isotopes. Following the same procedure as Stinnett [91], we simulated the same sources using the same 20s measurement time. Simulated sources were modeled after measured sources that are included in the AIP. The sources include: ^{241}Am, ^{133}Ba, ^{57}Co, ^{60}Co, ^{137}Cs, ^{152}Eu, ^{67}Ga, ^{125}I, ^{131}I, ^{192}Ir, ^{40}K, ^{226}Ra, ^{99m}Tc, ^{201}Tl, ^{233}U, high enriched uranium (HEU), and weapons grade plutonium (WGPu). HEU was defined to be identified correctly if the ANN's highest posterior probability output identified ^{235}U. WGPu was defined to be identified correctly if the ANN identified either ^{241}Am, ^{239}Pu, or ^{240}Pu. Each of these isotopes are present in WGPu spectra.

F1 scores for trained ANNs and the peak-based Bayesian classifier are shown in Table 4.5. As seen in Table 4.5, each ANN, except the simple autoencoders, outperform the Bayesian classifier. Complete ANNs match or outperform their simple counterparts. The complete DNN achieves the highest F1 score.

Table 4.5: F1 scores from the Bayesian classifier and ANNs. The highest F1 scores are shown in bold.

Model	**F1 Score** [%]
Bayesian classifier	71.47
Simple CNN	78.79
Simple DNN	72.73
Simple CAE	66.67
Simple DAE	60.61
Complete CNN	78.79
Complete DNN	**84.85**
Complete CAE	78.79
Complete DAE	78.79

Incorrect identifications are shown in Table 4.6. Incorrect identifications typically involve isotopes that present only a few low energy photopeaks. Examples of this are ^{125}I classified as ^{241}Am; ^{241}Am classified as ^{204}Tl and ^{133}Xe; ^{99m}Tc classified as ^{131}I; and ^{57}Co classified as ^{99m}Tc. The Bayesian classifier also experienced issues identifying isotopes in this category, particularly ^{125}I. The Bayesian classifier also misidentified ^{233}U, likely due to contamination from ^{232}U.

Table 4.6: Isotopes incorrectly classified by the ANNs. Simple and complete ANNs are indicated by the S- and C- prefix respectively. Entries with dashes indicate a correct identification.

True Label	**S-CNN**	**S-DNN**	**S-CAE**	**S-DAE**
^{125}I	^{241}Am	^{241}Am	^{241}Am	^{241}Am
^{233}U	^{137}Cs	^{226}Ra	^{57}Co	^{226}Ra
WGPu	-	^{201}Tl	^{201}Tl	^{201}Tl
^{241}Am	-	^{204}Tl	-	^{133}Xe
^{40}K	Background	^{60}Co	Background	^{60}Co
^{57}Co	-	-	-	^{60}Co
^{99m}Tc	^{131}I	^{131}I	^{131}I	^{131}I

True Label	**C-CNN**	**C-DNN**	**C-CAE**	**C-DAE**
^{125}I	^{241}Am	^{241}Am	^{241}Am	^{241}Am
^{233}U	^{99}Mo	^{137}Cs	^{153}Sm	^{137}Cs
^{235}U	-	-	^{177m}Lu	-
^{241}Am	^{133}Xe	-	-	-
^{57}Co	-	-	-	^{99m}Tc

4.5 Chapter Discussion and Conclusion

This chapter shows that convolutional models performed well when identifying spectra in both simulated and measured data. When identifying simulated data in Section 4.2, the CNN's and CAE's often displays comparable performance. When identifying real spectra in Section 4.3, the simple CAE often achieves a larger posterior probability for the correct isotope compared to the simple CNN. This shows that the CAE's pretraining allows the convolutional architecture to better identify real spectra who's parameters are outside the training dataset. With additional simulated parameters in the complete training dataset, the CNN outperforms the CAE in real spectra.

This chapter also shows that the posterior probabilities of complete ANNs converge to an asymptote quicker and more monotonically than the simple ANNs. These features are desirable for an isotope identification algorithm because they help untrained users interpret the identification results. For

example, a sudden increase in an identified isotope's posterior probability could indicate a source is moving towards the detector, increasing the signal-to-background ratio of the spectrum. If the ANN's posterior predictions are erratic, increases and decreases in posterior probability would be meaningless.

Finally, this chapter demonstrated an improvement over a peak based Bayesian classifier designed to perform isotope identification using the same isotope library as our work. Each complete ANN outperformed the Bayesian classifier by at least 7 points on an F1 score measured using the same dataset.

CHAPTER 5

URANIUM ENRICHMENT REGRESSION RESULTS AND DISCUSSION

Measuring uranium enrichment is difficult for a number of reasons. First, characteristic ^{235}U gamma-rays are easily shielded. Second, restriction may be placed on inspectors performing measurements for treaty verification. Inspectors can face two major restrictions: a limited amount of time to measure data and the requirement for an information barrier. An information barrier is anything that restricts the inspector from measuring information outside of their specific target. For example, if an inspector measured the gamma-ray spectrum of an object using a high-resolution HPGe detector, they could extract information about the process used to process used to bred and enrich the material. This information could be a state secret, and thus would need to be protected with some barrier. The low-resolution inherent to NaI(Tl) is an information barrier for treaty verification measurements. An addition information barrier is an ANN which is trained to only extract enrichment information.

This chapter demonstrates a validation demonstration applying `annsa` to automated uranium enrichment measurements using NaI(Tl). In this chapter we discuss how we used MCNP and GADRAS-DRF to create spectral templates for training dataset simulation with `annsa`. We also train and benchmark ANNs on simulated and measured enriched uranium spectra measured at the Nevada National Security Site and as part of a IAEA training exercise.

5.1 Uranium Enrichment Measurement Background

Verifying the enrichment of HEU through passive nondestructive analysis is important for nuclear safeguards applications and homeland security tasks. Passive nondestructive analysis, such as gamma-ray spectroscopy, is preferred to more accurate destructive methods due to its ability to operate quickly,

preserve forensic evidence, and allow for remote measurements. The enriched uranium gamma-ray signatures visible in low-resolution detectors come from the decay of ^{238}U, ^{235}U, ^{232}U, and the daughters of these isotopes. Figure 5.1 shows an example of a 27% enriched uranium spectrum that displays ^{238}U and ^{235}U photopeaks. In low-resolution detectors, The primary photopeaks from ^{235}U are at 144 keV, 163 keV, and 186 keV. The ^{238}U daughter ^{234m}Pa produces two main photopeaks at 766 keV and 1001 keV. ^{232}U, only present in reprocessed uranium, produces a photopeak at 2.6 MeV from its ^{208}Tl daughter.

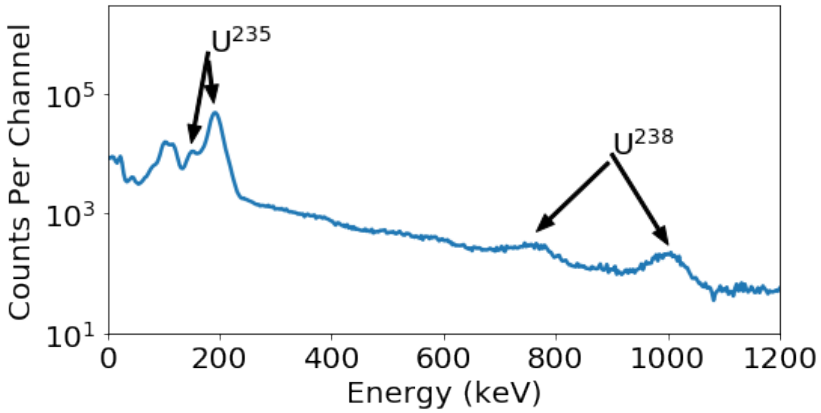


Figure 5.1: 27% enriched uranium spectrum measured with a NaI(Tl) detector.

.

Traditionally, the enrichment meter method is used to measure uranium enrichment in NaI(Tl) spectra [92]. This, and all enrichment measurement methods based on gamma-ray spectroscopy, only measures the enrichment of an object's surface to a depth of 0.26 cm and 0.74 cm for uranium metal and U_3O_8 powder, respectively [93]. At these material depths, the attenuation properties of uranium removes 99.8% of the 186 keV signal. The enrichment meter method method exploits the proportionality between the activity of the 186 keV photon and the enrichment of ^{235}U. The enrichment meter method works by finding calibration constants that relate counts in two ROIs,

$$E = AC_{ROI1} + BC_{ROI2} \tag{5.1}$$

where

$$C_{ROI1} = \text{counts in ROI1 [\# counts]}$$
$$C_{ROI2} = \text{counts in ROI2 [\# counts]}$$
$$A = \text{calibration constant}$$
$$B = \text{calibration constant,}$$

to the measured material's enrichment. These ROIs are placed around the 186 keV peak and in the background region to the right of the peak. Finding these constants requires measuring two different uranium enrichment standards in the same configuration (shielding, scattering environment, source-detector distance). Once these calibration constants are measured, they can only be used in the same configuration as the reference standards. It is possible to extend this method to other shielding configurations, but this introduces a risk of adding systematic errors. An automated version of this method called NaIGEM (NaI(Tl) Gamma Enrichment Measurements) is included in the HM-5 instrument used by the IAEA [94]. Enrichment measurements of uranium without contaminants using low-resolution detectors can achieve 1% precision for arbitrary enrichments while contamination by minor uranium isotopes have a biasing effect of 5-10% [95]. Measurements of materials with unknown enrichment, shielding, and geometry are typically performed with high resolution HPGe detectors using the Multi-Group Analysis for Uranium (MGAU) software [96]. Using a NaI(Tl) detector to measure the enrichment of an item without knowing these parameters is inherently challenging due to the low-resolution of NaI(Tl). By training ANNs with a dataset of simulated enriched uranium spectra, it may be possible to use this material to perform uranium enrichment measurements.

5.2 Problem Description and Training Dataset Overview

To investigate how well a machine learning algorithm can learn to perform uranium enrichment measurements, machine learning architectures found in Chapter 3 were trained using a dataset of simulated enriched uranium spec-

tra.

5.2.1 Training Outline

Simple and complete model architectures discussed in Sections 3.6 and 3.8 were trained using dataset of 10^5 simulated uranium spectra. Model architectures were modified to perform regression. Modifications include changing: the number of output nodes to one, their output function to a sigmoid, and their main cost function to mean squared error. Typically the sigmoid output function is used in the context of logistic regression. In our case the regression targets, percent ^{235}U enrichment, exist on [0,1], allowing use of the sigmoid output function. Using this function, we interpret each ANN's output as an enrichment value on [0,1] instead of a list of isotope posterior probabilities. Pretrained simple and complete DAE and CAE models from Section 3.4.3 were also fine-tuned using the simulated uranium dataset. Each model was trained using 5-folds cross validation. For each test dataset in this chapter, uranium enrichment predictions from each trained model were averaged to reduce the prediction's variance. This process is illustrated in Figure 5.2.

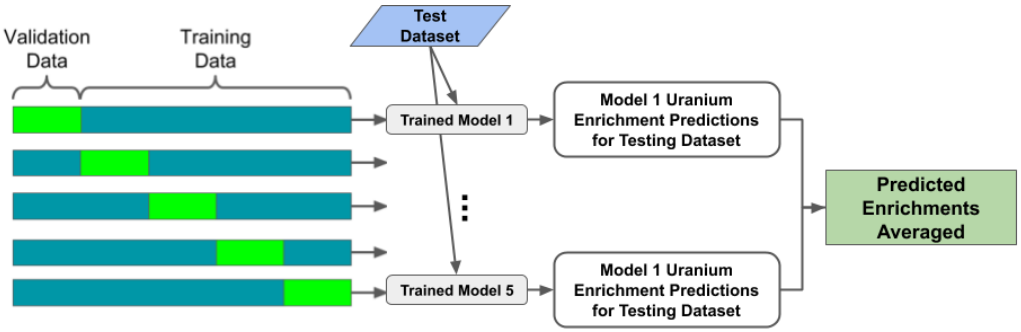


Figure 5.2: Training and prediction averaging.

5.2.2 Training Dataset Details

A coupled MCNP (Monte Carlo N-Particle Transport Code) and GADRAS-DRF simulation approach was used to simulate the spectrum from each uranium isotope uniformly distributed in a solid uranium sphere of radius 5.5 cm. MCNP simulation was used to calculate the physics due to self attenuation in the uranium and GADRAS-DRF was used to model the gamma-ray

spectrum from this source in a 2" x 2" NaI(Tl) detector. The universe in the MCNP simulation, shown in Figure 5.3, was empty except for a 19 cm concrete block fixed at 180 cm from the origin, a 5.5 cm radius sphere of bare uranium fixed at the origin, and a bare 2" x 2" NaI(Tl) cylinder located between them. The concrete block was added to incorporate backscatter radiation in the MCNP simulation. A total of 10^8 particles were simulated for each configuration.

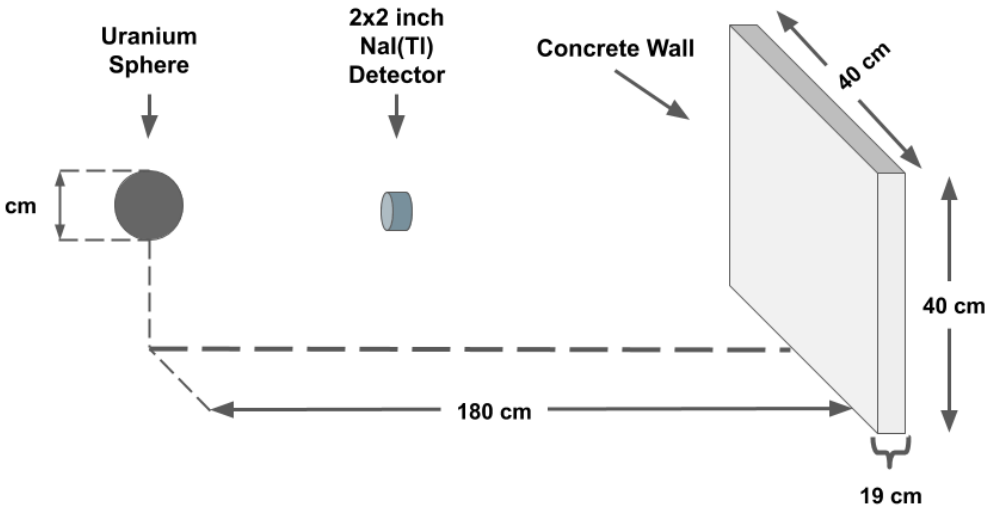


Figure 5.3: Diagram of MCNP simulation (not to scale).

The software package RadSrc was used to generate specific gamma-ray intensities for the ^{235}U, ^{238}U, and ^{232}U templates [97]. RadSrc, developed at Lawrence Livermore National Laboratory, uses the Bateman equations to calculate daughter in-growth and their respective specific gamma-ray intensities. Isotopes in enriched uranium reach secular equilibrium in about six months. To account for this, RadSrc was used to find specific gamma-ray intensities for 50 year old uranium isotopes and their ingrown daughters. The specific activities for each isotope are found in Table 5.1.

Table 5.1: Specific activities for 50 year old uranium isotopes and their ingrown daughters.

Isotope	**Specific Gamma-ray Activity** $[\frac{\gamma}{gs}]$
^{232}U	1.20 x 10^{12}
^{235}U	2.07 x 10^{5}
^{238}U	3.80 x 10^{3}

^{235}U and ^{238}U templates were combined in weighted combinations to create spectra of desired enrichments using the process described previously in Equation 2.31. To account for factors that change the ^{232}U content, each sample that included ^{232}U used a mass fraction uniformly chosen between 4 x 10^{-9} and 2 x 10^{-8} [98]. To account for clean uranium, the probability that a spectrum was simulated with ^{232}U was 0.5. Complete simulation parameters for the coupled MCNP and GADRAS-DRF approach are shown in Table 5.2.

Table 5.2: Range of parameters used for the uranium enrichment dataset.

Simulation Parameter	**Range**	**Sampling**
Source-Detector Distance [cm]	25, 50, 100, 150, 200	Uniform
FWHM at 662 keV [%]	6.5, 7.0, 7.5	N/A
Shielding (% 200 keV Attenuated)	0%, 20%, 40%, 60%	Uniform
Integration Time [s]	60 - 3600	Log-Uniform
Calibration Offset [channels]	-10 - 10	Uniform
Calibration Gain	0.8 - 1.2	Uniform
^{235}U Enrichment [%]	0 - 100	Uniform
Background Counts per Second	170 - 230	Uniform
Signal to Background Ratio	1.0 - 4.0	Log-Uniform

5.3 Results - Simulated Data

These sections describe how each trained model performs when predicting the uranium enrichment of simulated spectra. To investigate performance, spectra were simulated with enrichments of 3%, 25%, 50%, 75%, and 93% in different conditions. To observe each ANN's enrichment prediction variance, 10 spectra were simulated for each enrichment value. Each ANN's mean and variance when identifying all spectra at each enrichment value was recorded. Each ANN used the model averaging technique shown in Figure 5.2. Default simulation parameters are shown in Table 5.3. Changes to these defaults are indicated for each generalization experiment.

Table 5.3: Default parameters used for all generalization datasets.

Simulation Parameter	**Value**
Source-Detector Distance [cm]	50.0
FWHM at 662 keV [%]	7.0
Shielding (% 200 keV Attenuated)	0%
Integration Time [s]	600.0
Calibration - Offset (channels)	0.0
Calibration - Gain	1.0
Signal to Background Ratio	3.0
Background Counts Per Second	200.0

5.3.1 Effect of Shielding on Enrichment Prediction

To test each model's ability to measure uranium enrichment when the uranium source is shielded, enriched uranium spectra were simulated with various thicknesses of shielding. The effect of each shielding thickness is shown in Figure 5.4. The predicted enrichments from each model on these cases are shown in Figure 5.5. In general for each case, the complete networks outperform the simple networks by predicting enrichments closer to the simulated value. This indicates that the reduced capacity of the simple networks is insufficient to properly perform uranium enrichment measurements using the provided dataset. At 93% enrichment each model underestimates the enrichment.

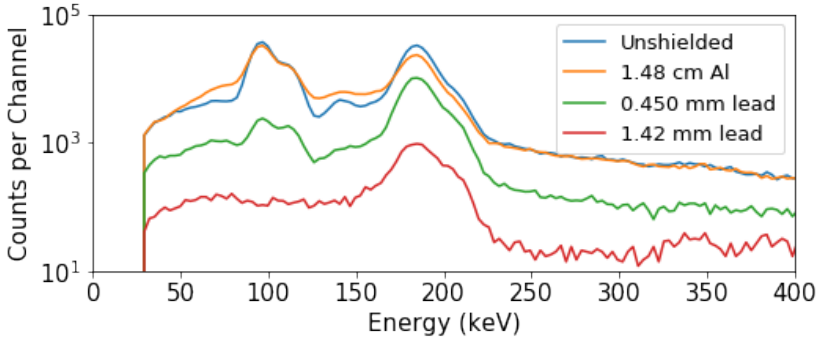
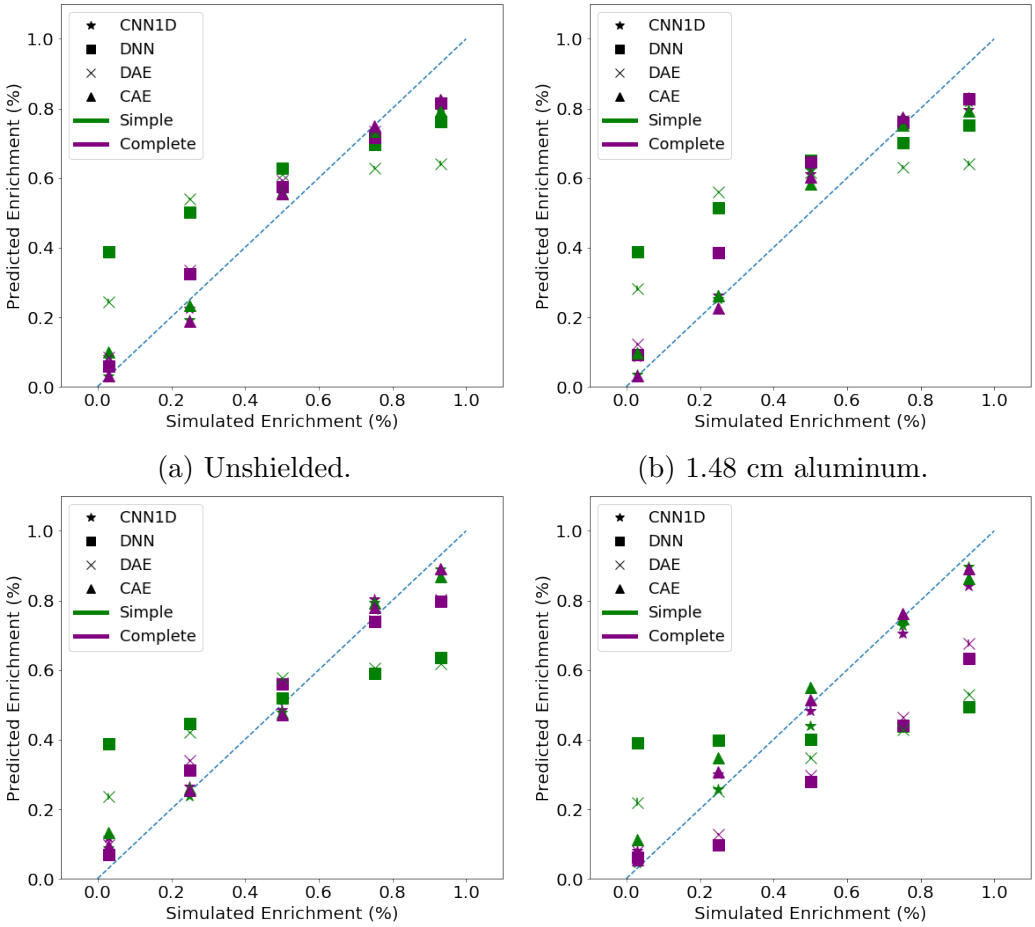


Figure 5.4: Simulated 93% enriched uranium spectra various amounts of shielding.



(a) Unshielded.

(b) 1.48 cm aluminum.

(c) 0.450 mm lead.

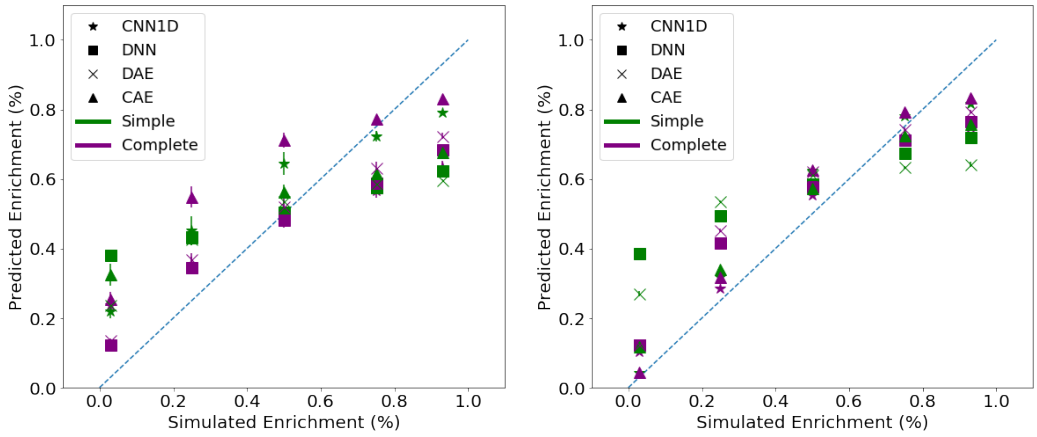
(d) 1.42 mm lead.

Figure 5.5: Each model's prediction for the uranium enrichment of spectra simulated with different shielding thicknesses. Shielding conditions are indicated below each figure.

In all cases except with 1.42 mm of lead shielding, simple networks overestimate the enrichment at and below 50%. The simple models also underestimate the enrichment at values over 50%. This reinforces the conclusion that the simple network has too little capacity to fit the data. Because the range of enrichments in the training data are uniformly distributed from 0 - 100%, the naive method to minimize the cost function is to predict enrichments near 50%. The complete models were within 20% of the true value in all cases except with 1.42 mm of lead shielding. This shows that complete models are only affected by relatively large amounts of shielding.

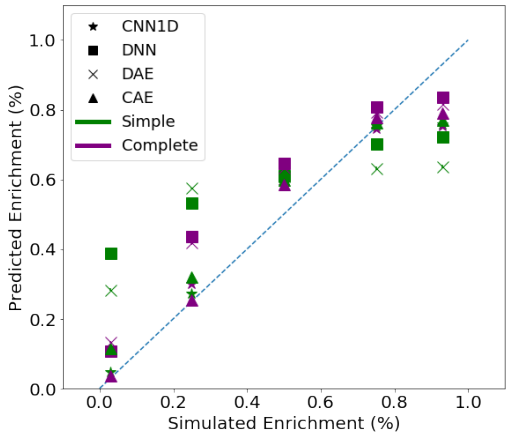
5.3.2 Effect of Calibration on Enrichment Prediction

In this section we test each model's ability to measure uranium enrichment in spectra with various relative calibration gains. The predict enrichments from each model on these cases are shown in Figure 5.6. Model performance at relative gains of 0.9 and 1.1 are similar to the reference gain case, shown in Figure 5.5a.

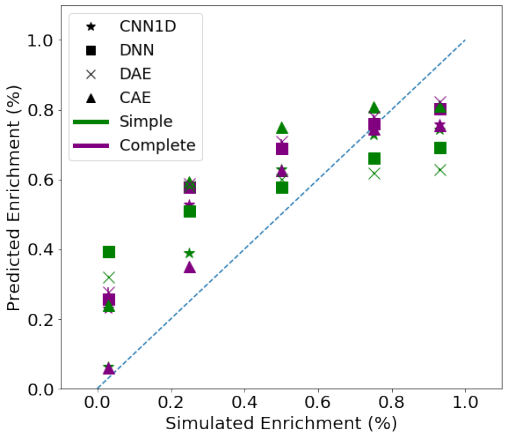


(a) 0.8

(b) 0.9



(c) 1.1



(d) 1.2

Figure 5.6: Each model's prediction for the uranium enrichment of simulated spectra calibrated with different gain settings. The magnitude of the applied relative gain shift are shown below each figure.

At the extreme relative gain shifts of 0.8 and 1.2 each ANN's performance visibly changes compared to the reference gain. These shifts represent large deviations in normal NaI(Tl) operating temperature or a significantly mis-calibrated detector. At a relative gain shift of 0.8, the dense models are

not visibly impacted compared to the reference gain case when measuring uranium with enrichments below 75%.

At a gain shift of 1.2 each ANN overestimates enrichment for enrichments at and below 50%. The complete CAE and simple CNN are the most accurate when predicting enrichment values at and below 25%. Because the 1.2 relative shift moves the 766 keV and 1001 keV ^{238}U peaks into regions of little importance for the spectra without gain shift, seen in Figure 5.8, these peaks are ignored and the enrichment is overestimated.

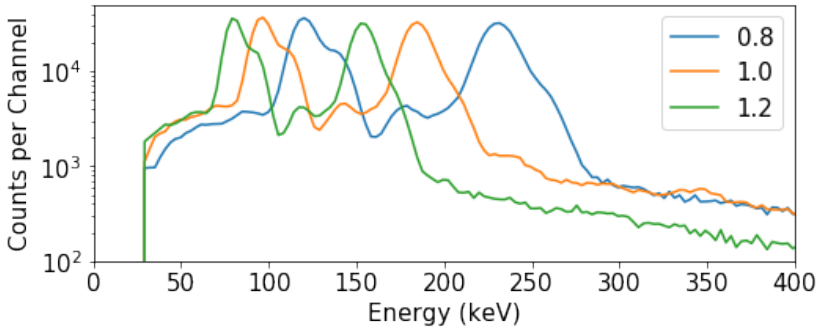


Figure 5.7: Simulated 93% enriched uranium spectra with three different relative gain settings. Energy calibration shown is based on the 1.0 relative gain setting.

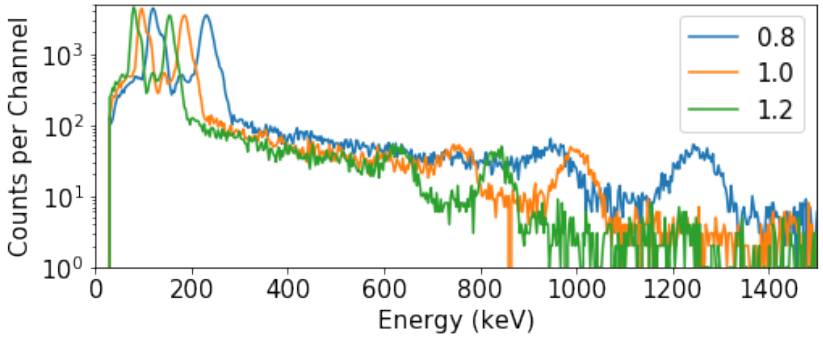


Figure 5.8: Simulated 10% enriched uranium spectra with three different relative gain settings. Energy calibration shown is based on the 1.0 relative gain setting.

5.4 Results - Measured Spectra

To quantify performance in real gamma-ray spectra, material with different uranium enrichments were measured with a variety of NaI(Tl) detectors. Ob-

jects measured include a 13 kg 93% enriched uranium metal sphere [99] measured at the Nevada National Security Site and three U_3O_8 samples measured as part of an IAEA training exercises [100]. Shielding, source-detector distance, radiation background, and scattering environments were not recorded for the training exercise. The U_3O_8 spectra were measured with energies below 1.3 MeV. Energies above this threshold do not include information about the object's enrichment, so removing it will have negligible effect on the ANN predictions. In addition to this, each detector used to measure the U_3O_8 spectra had a unique calibration. We manually recalibrated each spectrum using gain and offset correction to match each 186 keV and 1001 keV photopeak. These recalibrated spectra are shown in Figure 5.9.

Table 5.4: Uranium sample description.

Enrichment	^{235}U Mass (g)	Material	Live Time (s)
93.1%	13,000	U Metal	300
91.4%	903	U_3O_8	226
27.1%	264	U_3O_8	344
0.7%	-	U_3O_8	97.4

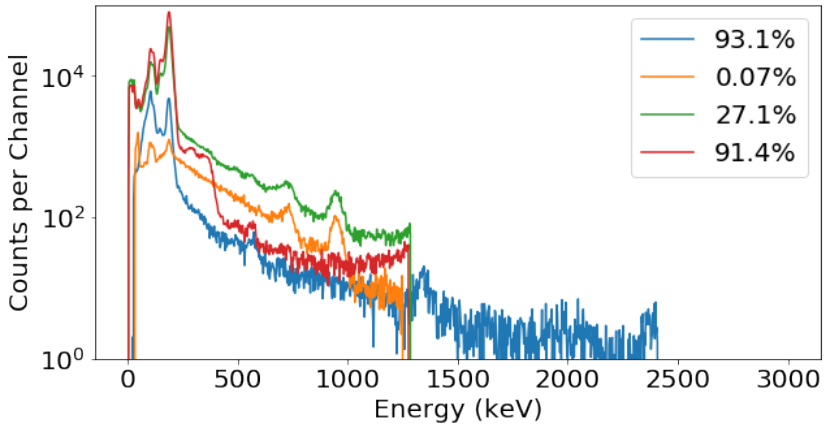


Figure 5.9: Enriched uranium spectra recalibrated to the 186 keV and 1001 keV photopeaks.

Figure 5.10 shows the predicted enrichment's mean and variance for each model and measured spectrum. Similar to the simulated spectra, the simple architectures underestimate enrichments over 50% and overestimate enrichments under 50%. Each network's prediction variance was not visible. In

general, the complete architectures were more accurate than the simple architectures.

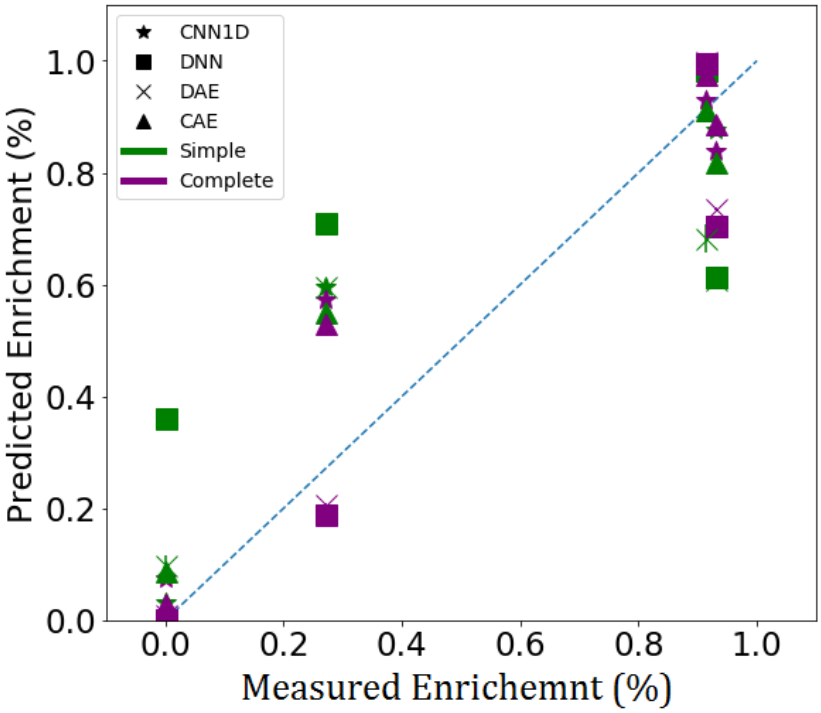
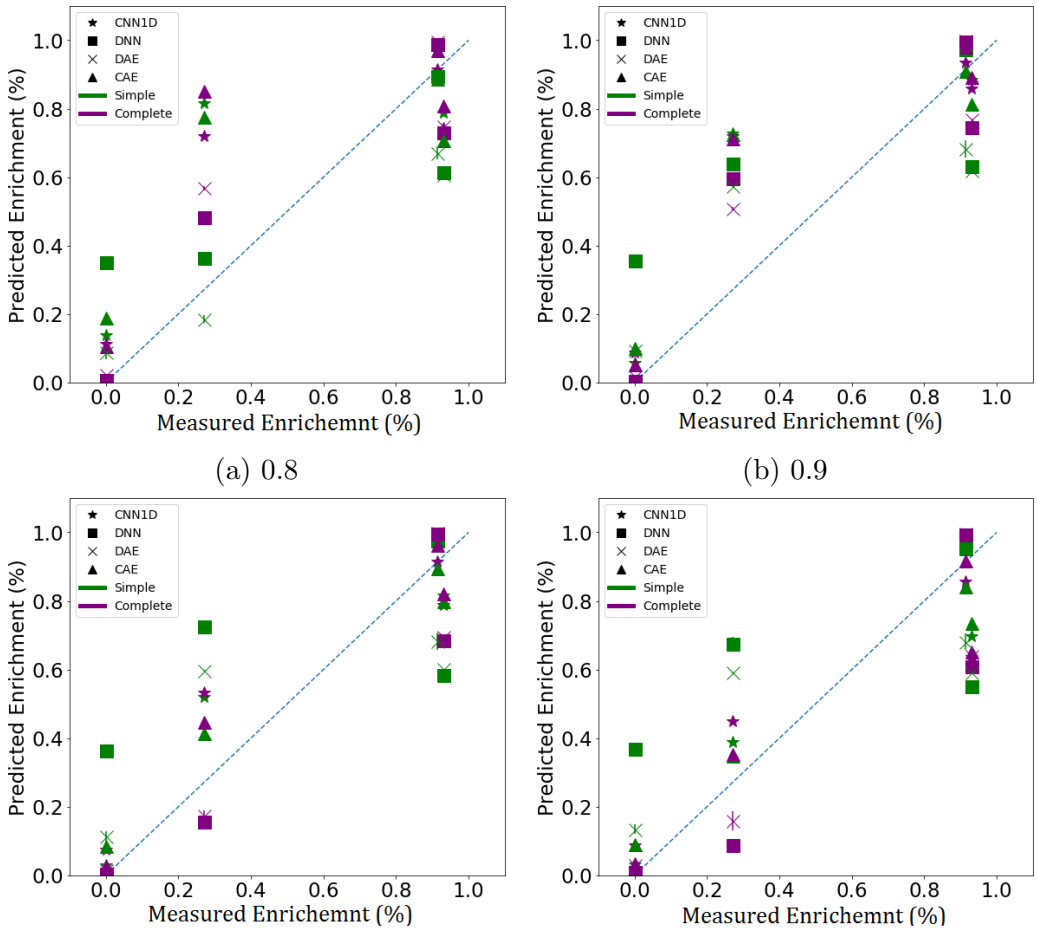


Figure 5.10: Each model's predicted uranium enrichment for measured uranium spectra.

At 27.1% enrichment, most ANNs overestimate the enrichment by at least 13 points. The complete dense models more accurately predict enrichment values for the 27.1% enriched uranium spectrum. Spectra with lower enrichment include a continuum-like signal from an effect called bremsstrahlung. Bremsstrahlung (German for "breaking radiation") is produces when charged particles accelerate in the electric field of a heavy nucleus. In enriched uranium, bremsstrahlung is produced from the 2.3 MeV beta particle that is associated with the decay of ^{234m}Pa, a ^{238}U daughter. Convolutional models extract features based on their 16 channel filter kernel lengths. Without the bremsstrahlung signal in the training dataset, convolutional models have learned feature extraction techniques that cannot work when the bremsstrahlung signal is present in low-enriched data. Dense models are forced to extract features in patches of 16 channels, so they can more easily ignore the bremsstrahlung continuum and use the ^{235}U and ^{238}U photopeaks for enrichment prediction.

To investigate how changing calibration effects each algorithm, the spectra from Figure 5.9 were recalibrated with the same gain settings explored in Section 5.3.2. Each ANN's enrichment prediction for these spectra is shown in Figure 5.11.



(a) 0.8 (b) 0.9

(c) 1.1 (d) 1.2

Figure 5.11: Each model's prediction for the uranium enrichment of measured spectra calibrated with different gain settings. The magnitude of the applied relative gain shift are shown below each figure.

Similar to the simulated results, relative gain settings of 0.9 and 1.1 do not significantly impact enrichment quantification for most networks. The relative gain setting of 0.9 increases the dense model's enrichment prediction for the 27% enriched spectrum. In general, models overestimate the enrichment of 0.1% and 27% enriched spectra using the relative gain setting of 0.8. In all gain settings, ANNs underestimate 93% enriched uranium spectrum's enrichment.

5.5 Discussion and Conclusion

The ability to distinguish low (<20% ^{235}U)and high (>85% ^{235}U) enriched uranium is extremely important. Low enriched uranium requires significant processing for use in a nuclear weapon while high enriched uranium requires much less processing. The current implementation can roughly differentiate between low and high enriched uranium. For example, it is reasonable to assume a sample of material is below 15% enrichment if the complete DAE predicts an enrichment below 15%. Similarly, complete DAE predictions above 25% can be assumed to be high enriched uranium. Predictions within this range require additional testing to more accurately quantify enrichment.

This chapter shows that complete architectures outperform the simple architectures when predicting uranium enrichment. The results also show that the complete architectures tailored to isotope classification can be extend to other problems in gamma-ray spectroscopy. The chapter also demonstrates that autoencoders trained to reconstruct spectra for isotope identification can be used to pretrain networks for other spectroscopic tasks. The dense pretraining was shown to perform particularly well on both simulated and measured enriched uranium spectra. Dense models show promise in uranium enrichment regression. This is because the feature extraction processes used by the convolution architectures are tuned to isotope identification. The convolutional models used convolutional filters with lengths of 16 channels. Uranium enrichment regression may require shorter convolution filter lengths to extract features from photopeaks below 200 keV.

International safeguards measurements require high precision measurements in ranges similar to those offered by the enrichment meter method. To expand our presented method for use in international safeguards, the accuracy needs to be improved and tested on additional enriched uranium spectra. Accuracy can be improved by expanding the training dataset to include additional facility measurement scenarios. For measurement in nuclear processing facilities, additional scattering scenarios that mimic heavy equipment and small uranium samples that cannot be assumed to be infinitely thick at 186 keV must be added. The ^{232}U, ^{234}U, ^{233}U, ^{236}U, and bremsstrahlung contribution can also be modeled more accurately, especially to use this method in higher-resolution detectors.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 General Conclusions

This dissertation demonstrates a method to optimize machine learning architectures for an isotope classification task and a uranium enrichment regression task using low-resolution gamma-ray spectroscopy. This work demonstrates a clear contribution to the field of radioisotope identification via gamma-ray spectroscopy. In this work we introduce `annsa`, an open source Python package designed to facilitate machine learning for gamma-ray spectroscopy tasks.

Using `annsa`, we analyzed convolutional and dense neural network architectures and delivered insights into how hyperparameters affect each model's performance. In an attempt to shed light on a black box, hyperparameters from random searches were analyzed with respect to the validation dataset's F1 score. Future experiments into machine learning architectures for gamma-ray spectroscopy should be guided by the hyperparameter bounds and discussion from this work.

Analysis from the hyperparameter search shows that dense ANNs trained using both the simple and complete training dataset obtain the best validation dataset F1 score with fewer dense layers. This shows that additional dense layers produce models susceptible to overtraining. Overtraining was also observed when the total number of dense nodes increased above 1 x 10^3. Models trained with the complete dataset show improved performance as total dense nodes increased to this limit. We also found that preprocessing by taking the *sqrt* of the spectrum resulted in the best validation dataset F1 score for the simple and complete dense models.

Convolutional ANNs required longer filter lengths and additional convolutional layers for best performance. The complete CNNs need longer filters

than the simple CNNs for good performance on their respective validation datasets. Longer filters are needed to synthesize information in a larger area of the spectrum. Complete CNNs also required longer pooling sizes. Longer pooling sizes increase model invariance to the gain changes present in the complete training dataset.

This work also demonstrates a general method to construct training datasets for spectroscopic tasks available in `annsa`. We demonstrate this dataset construction method by creating training datasets for radioactive source interdiction and uranium enrichment regression.

In most tested source interdiction cases, we show that convolutional models outperform dense models by achieving higher F1 scores on simulated testing datasets or by achieving higher correct posterior probabilities on measured sources. Experiments were created to test how each ANN's performance was effected by changing simulated and physical parameters that distort gamma-ray spectra. These experiments were performed for two source interdiction training datasets, the first composed of a narrow range and the second composed of a wider range of simulated parameters. From these experiments we observe that convolutional models show the most promise for source interdiction, often achieving the highest F1 score on simulated datasets and high posterior probabilities for correct isotopes in measured data. Convolutional models perform better because they assume local structure in the spectrum while dense models make no assumption about spectral structure. Overall, machine learning models show good performance on a wide range of detector calibrations when tested against both simulated and measured spectra. This performance makes machine learning models good candidates for use on handheld RIID detectors where the calibration is often unreliable.

We also observe that including shielding in the classification training dataset is necessary to correctly identify shielded isotopes when tested against both simulated and measured spectra. Autoencoders trained without shielding demonstrate some generalization capabilities when identifying measured shielded spectra. The pretrained ability to reconstruct spectra uniquely benefits the autoencoders when identifying distorted spectroscopic signals and should be investigated further for practical applications. In measured spectra, we show that including shielding in the training dataset is important even for isotopes with simple spectral signals.

The work demonstrates that the machine learning architectures found from

the hyperparameter search tuned to a classification task can be extended to other spectroscopic tasks such as quantifying uranium enrichment. Using these architectures and a dataset constructed to perform uranium enrichment quantification, we demonstrated ANNs that can differentiate between low and high enriched uranium in measured gamma-ray spectra. Because of the greater threat posed by high enriched uranium versus low enriched uranium, differentiating these two classes using low-resolution detectors is incredibly important.

While CNNs outperform DNNs in source interdiction tasks, dense models show better performance when quantifying uranium enrichment in measured spectra. This shows that the convolutional kernel sizes are inappropriately large for features in enriched uranium spectra or that CNNs are better suited to the pattern recognition problem of isotope identification. This also shows that dense networks are better suited for quantification tasks which require comparing counts in specific spectral regions. Fundamentally, this demonstrates that a machine learning model must be carefully chosen and tailored to a given spectroscopic task.

6.2 Suggested Future Work

To implement ANNs in a high performance commercial detection system, a few improvements are required. Dataset simulation parameters should more accurately reflect real-world conditions. To achieve this, a more realistic background count-rate distribution and additional background templates should be added to the training dataset. Devices should also come with networks designed for specific background environments, such as the background expected in a certain city or in different geological areas. Modeled source strengths should be based on expected count rates from each source in a range of activities expected. Additional simulated NaI crystal variations would need to be added to the training set because manufacturing differences between NaI shape effects can affect peak-to-total ratios and detector intrinsic efficiency. Shielding should be added based on how much information content is lost from the source signal when increasing shielding is added.

In order to ensure accuracy appropriate for a commercial system, additional validation datasets must be investigated. Validation datasets are

needed to investigate if certain isotope combinations could mask each other or otherwise change identification. Validation datasets are also necessary to test each algorithm's detection limits with respect to source strength and measurement time. These datasets can be used to tailor alarm thresholds for operational requirements.

A simulated training dataset allows the same machine learning model optimization process to be applied for different detector materials like plastic scintillator, CZT, and HPGe detectors. Different detector materials produce spectra with unique features and significantly different resolutions. The hyperparameter selection process outlined in this work can be repeated for these materials and optimum hyperparameters can be compared. This will help us understand how each machine learning model uses spectral features.

In addition to the deep learning algorithms presented in this thesis, more classical machine learning algorithms such as the support vector machine, random forests, and k-means clustering should be applied to similar datasets an their performance analyzed. Performance could also be compared to models trained on feature extraction methods like autoencoders or principal component analysis.

Pretrained autoencoders could be further explored for a variety of applications. Autoencoders pretrained using low-resolution NaI spectra could be used to train networks for detectors with different resolution. Different autoencoder feature extraction processes can also be explored. Examples include: using a spectrum as input and outputting posterior probability that each channel contains a photo-peak and using a gain-shifted spectrum as input and outputting a gain-corrected spectrum. The encoding from different feature extraction methods could be fed into a DNN, support vector machine, random forests, or k-means algorithm and their performance compared.

REFERENCES

[1] N. U. Security and T. Laboratory, "Handheld radionuclide identification devices (rids) market survey report," 9 2015. [Online]. Available: https://www.dhs.gov/sites/default/files/publications/HHRID-MSR_0915-508.pdf

[2] G. Knoll, *Radiation Detection and Measurement*, 4th ed. John Wiley and Sons, 2010, ISBN: 978-0470131480.

[3] G. R. Gilmore, *Practical Gamma-ray Spectrometry*, 2nd ed. John Wiley and Sons, 2008, ISBN: 978-0471951506.

[4] J. Kulisek, J. Schweppe, S. Stave, B. Bernacki, D. Jordan, T. Stewart, C. Seifert, and W. Kernan, "Real-time airborne gamma-ray background estimation using nasvd with mle and radiation transport for calibration," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 784, pp. 287 – 292, 2015, symposium on Radiation Measurements and Applications 2014 (SORMA XV). [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900214014247

[5] USGS, "Open-file report 2005-1413: Terrestrial radioactivity and gamma-ray exposure in the united states and canada," http://pubs.usgs.gov/of/2005/1413/maps.htm, 2013.

[6] R. Casanovas, J. Morant, and M. Salvad, "Temperature peak-shift correction methods for nai(tl) and labr3(ce) gamma-ray spectrum stabilisation," *Radiation Measurements*, vol. 47, no. 8, pp. 588 – 595, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1350448712001692

[7] M. Moszyski, A. Nassalski, A. Syntfeld-Kauch, T. Szczniak, W. Czarnacki, D. Wolski, G. Pausch, and J. Stein, "Temperature dependences of labr3(ce), lacl3(ce) and nai(tl) scintillators," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 568, no. 2, pp. 739 – 751, 2006. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016890020601196X

[8] M. Kamuda, J. Stinnett, and C. J. Sullivan, "Automated isotope identification algorithm using artificial neural networks," *IEEE Transactions on Nuclear Science*, vol. 64, pp. 1858 – 1864, 2017.

[9] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.

[10] W. contributor Chervinskii, "Autoencoder structure," 2015. [Online]. Available: https://commons.wikimedia.org/wiki/File:Autoencoder_structure.png

[11] R. Hofstadter, "Alkali halide scintillation counters," *Phys. Rev.*, vol. 74, pp. 100–101, Jul 1948. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.74.100

[12] L. Pibida, M. Unterweger, and L. R. Karam, "Evaluation of handheld radionuclide identifiers," *Journal of Research of the National Institute of Standards and Technology*, vol. 109, pp. 451–456, 2004.

[13] J. M. Blackadar, S. E. Garner, J. A. Bounds, W. H. Casson, and D. J. Mercer, "Evaluation of commercial detectors," *Proceedings of the INMM Southwest Section Meeting*, 2003.

[14] J. Blackadar, C. Sullivan, B. Rees, S. Garner, and D. Mercer, "Continuing evaluation of isotope identifiers," in *Proceedings of INMM 45th Annual Meeting, Orlando, FL*, 2004.

[15] T. Burr and M. Hamada, "Radio-isotope algorithms for NaI gamma spectra," *Algorithms*, vol. 2, pp. 339–360, March 2009.

[16] M. Swoboda, R. Arlt, V. Gostilo, A. Lupilov, M. Majorov, M. Moszynski, and A. Syntfeld, "Spectral gamma detectors for hand-held radioisotope identification devices (RIDs) for nuclear security applications," in *Nuclear Science Symposium Conference Record, 2004 IEEE*, vol. 7, 2004, pp. 4296–4302.

[17] M. Kamuda, "Automated isotope identification algorithm using artificial neural networks," Master's thesis, University of Illinois at Urbana-Champaign, 2017.

[18] M. Kamuda, J. Zhao, and K. Huff, "A comparison of machine learning methods for automated gamma-ray spectroscopy," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 1 2018.

[19] W. S. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin Of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[20] D. O. Hebb, *The Organization of Behavior*. John Wiley & Sons, Inc., 1949.

[21] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[22] F. Rosenblatt, *Principles of Neurodynamics*. Spartan Books, 1962.

[23] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[24] M. Minsky and S. A. Papert, *Perceptrons*. The MIT Press, 1969.

[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[26] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems 25*, pp. 1097–1105, 2012. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[28] S. Jeyanthia, N. Maheswaria, and R. Venkatesh, "Neural network based automatic fingerprint recognition system for overlapped latent images," *Journal of Intelligent & Fuzzy Systems*, vol. 28, 2015.

[29] I. J. Selvakumari Jeya and S. N. Deepa, "Lung cancer classification employing proposed real coded genetic algorithm based radial basis function neural network classifier." *Computational & Mathematical Methods in Medicine*, pp. 1 – 15, 2016.

[30] L. Hassan-Esfahani, A. Torres-Rua, A. Jensen, and M. McKee, "Assessment of surface soil moisture using high-resolution multi-spectral imagery and artificial neural networks." *Remote Sensing*, vol. 7, no. 3, pp. 2627 – 2646, 2015.

[31] A. Rababaah and S. D., "Integration of two different signal processing techniques with artificial neural network for stock market forecasting," *Academy of Information & Management Sciences Journal*, vol. 18, pp. 63–80, 2015.

[32] M. Rawool-Sullivan, J. Bounds, S. Brumby, L. Prasad, and J. Sullivan, "Steps toward automated gamma ray spectroscopy: How a spectroscopist deciphers an unknown spectrum to reveal the radioactive source," *LA-UR-10-01009*, February 2010.

[33] M. A. Mariscotti, "A method for automatic identification of peaks in the presence of background and its application to spectrum analysis," *Nuclear Instruments and Methods*, vol. 50, pp. 309–320, 1967.

[34] I. D. Lotto and A. Ghirardi, "Automatic peak-locating techniques for -ray spectra," *Nuclear Instruments and Methods*, vol. 143, no. 3, pp. 617 – 620, 1977. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0029554X77902579

[35] R. P. Gardner, X. Ai, C. R. Peeples, J. Wang, K. Lee, J. L. Peeples, and A. Calderon, "Use of an iterative convolution approach for qualitative and quantitative peak analysis in low resolution gamma-ray spectra," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 652, no. 1, pp. 544 – 549, 2011, symposium on Radiation Measurements and Applications (SORMA) XII 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900210030500

[36] H. Xiong, "Automated wavelet analysis of low resolution gamma-ray spectra and peak area uncertainty," Master's thesis, University of Illinois at Urbana-Champaign, 2015.

[37] J. Ely, R. Kouzes, J. Schweppe, E. Siciliano, D. Strachan, and D. Weier, "The use of energy windowing to discriminate SNM from NORM in radiation portal monitors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 560, no. 2, pp. 373 – 387, 2006.

[38] I. T. Jolliffe, *Principal Component Analysis*. Springer New York, 2002.

[39] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881–892, Jul 2002.

[40] V. V. Kumari, B. R. Raju, and A. Naik, "Hybrid clustering algorithm based on mahalanobis distance and mst," *International Journal of Applied Information Systems*, vol. 3, no. 5, pp. 60–63, Jul 2012.

[41] D. Boardman, M. Reinhard, and A. Flynn, "Principal component analysis of gamma-ray spectra for radiation portal monitors," *IEEE Transactions on Nuclear Science*, vol. 59, no. 1, pp. 154–160, Feb 2012.

[42] R. Runkle, "Analysis of spectroscopic radiation portal monitor data using principal components analysis," *IEEE Transactions on Nuclear Science*, vol. 53, no. 3, pp. 1418–1423, 2006.

[43] J. Mattingly and D. Mitchell, "A framework for the solution of inverse radiation transport problems," *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3734–3743, 2010.

[44] R. Abdel-Aal, "Comparison of algorithmic and machine learning approaches for the automatic fitting of gaussian peaks," *Neural Computing and Applications*, vol. 11, no. 1, pp. 17–29, 2002.

[45] R. Abdel-Aal and M. Al-Haddad, "Determination of radioisotopes in gamma-ray spectroscopy using abductive machine learning," *Nuclear Instruments and Methods in Physics Research A*, vol. 391, pp. 275–288, 1996.

[46] M. Medhat, "Artificial intelligence methods applied for quantitative analysis of natural radioactive sources," *Annals of Nuclear Energy*, vol. 45, pp. 73 – 79, 2012.

[47] V. Vigneron, J. Morel, M. Lepy, and J. Martinez, "Statistical modelling of neural networks in y-spectrometry," *Nuclear Instruments and Methods in Physics Research A*, vol. 396, p. 642647, 1996.

[48] V. Pilato, F. Tola, J. Martinex, and M. Huver, "Application of neural networks to quantitative spectrometry analysis," *Nuclear Instruments and Methods in Physics Research A*, vol. 422, pp. 423–427, 1999.

[49] L. Chen and Y.-X. Wei, "Nuclide identification algorithm based on KL transform and neural networks," *Nuclear Instruments and Methods in Physics Research A*, vol. 598, pp. 450–453, 2009.

[50] E. Yoshida, K. Shizuma, S. Endo, and T. Oka, "Application of neural networks for the analysis of gamma-ray spectra measured with a Ge spectrometer," *Nuclear Instruments and Methods in Physics Research A*, vol. 484, pp. 557–563, 2002.

[51] M. Kamuda, "annsa," https://github.com/arfc/annsa, 2019.

[52] D. J. Mitchell and L. T. Harding, "GADRAS isotope ID user's manual for analysis of gamma-ray measurements and api for linux and android," *SAND2014-3933*, May 2014.

[53] K. Ianakiev, B. Alexandrov, P. Littlewood, and M. Browne, "Temperature behavior of nai(tl) scintillation detectors," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*,

vol. 607, no. 2, pp. 432 – 438, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900209003416

[54] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, pp. 281–305, 2012.

[55] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no. 9, pp. 1423–1447, Sep 1999.

[56] R. Fletcher, *Newton-Like Methods*. John Wiley & Sons, Ltd, 2000, pp. 44–79.

[57] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference for Learning Representations*, 2015.

[58] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, pp. 251–257, 1991.

[59] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[60] X. Yu and G. Chen, "Efficient backpropagation learning using optimal learning rate and momentum," *Neural Networks*, vol. 10, p. 517527, 1997.

[61] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $o(1/k^2)$," in *Soviet Mathematics Doklady*, vol. 27, no. 2, 1983, pp. 372–376.

[62] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[63] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

[64] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[65] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, November 1998.

[66] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 142–150. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002491

[67] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009. [Online]. Available: http://www.cs.toronto.edu/ kriz/learning-features-2009-TR.pdf

[68] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," *Neurocomputing*, vol. 68, pp. 227–236, 1990.

[69] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[70] A. Zheng, *Evaluating Machine Learning Models*. O'Reilly Media, Inc., 2015.

[71] J. Lu, "Photopeak detection and quantification using wavelet analysis," Master's thesis, University of Illinois at Urbana-Champaign, 2013.

[72] C. J. Sullivan, S. E. Garner, K. B. Butterfield, and M. A. Smith-Nelson, "Wavelet analysis of gamma-ray spectra," *IEEE Nuclear Science Symposium Conference Record*, pp. 281–286, 2004.

[73] C. J. Sullivan, M. Martinez, and S. E. Garner, "Wavelet analysis of sodium iodide spectra," *IEEE Transactions on Nuclear Science*, pp. 2916–2922, 2006.

[74] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006. [Online]. Available: http://science.sciencemag.org/content/313/5786/504

[75] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Mar. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1756025

[76] D. Charte, F. Charte, S. Garca, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature

fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78 – 96, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253517307844

[77] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ser. ICML '08. New York, NY, USA: ACM, 2008, pp. 1096–1103. [Online]. Available: http://doi.acm.org/10.1145/1390156.1390294

[78] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, pp. 3371–3408, Dec. 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1756006.1953039

[79] M. Welling, "Fisher linear discriminant analysis," *Department of Computer Science*, vol. 16, 01 2007.

[80] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[81] H. Krekel, B. Oliveira, R. Pfannschmidt, F. Bruynooghe, B. Laugher, and F. Bruhin, "pytest 3.5," 2004. [Online]. Available: https://github.com/pytest-dev/pytest

[82] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[83] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 – 56.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[85] T. Goorley, M. James, T. Booth, F. Brown, J. Bull, L. J. Cox, J. Durkee, J. Elson, M. Fensin, R. A. Forster, J. Hendricks, H. G. Hughes, R. Johns, B. Kiedrowski, R. Martz, S. Mashnik, G. McKinney, D. Pelowitz, R. Prael, J. Sweezy, L. Waters, T. Wilcox, and T. Zukaitis, "Initial mcnp6 release overview," *Nuclear Technology*, vol. 180, no. 3, pp. 298–315, 2012. [Online]. Available: https://doi.org/10.13182/NT11-135

[86] S. Agostinelli *et al.*, "GEANT4: A Simulation toolkit," *Nucl. Instrum. Meth.*, vol. A506, pp. 250–303, 2003.

[87] Sandia National Laboratories, "Algorithm improvement program," 2016.

[88] *American National Standard Performance Criteria for Hand-Held Instruments for the Detection and Idenfication of Radionuclides, ANSI N42.34-2006*, IEEE Std., Rev. ANSI N42.34-2006, 2007.

[89] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," *CoRR*, vol. abs/1206.5533, 2012. [Online]. Available: http://arxiv.org/abs/1206.5533

[90] E. Browne and R. B. Firestone, *Table of Radioactive Isotopes*, V. S. Shirley, Ed. Wiley-Interscience Publications, 1986.

[91] J. Stinnett, "Automated isotope identification algorithms for low-resolution gamma spectrometers," Ph.D. dissertation, University of Illinois at Urbana-Champaign, 2016.

[92] T. Reilly, R. Walton, and J. Parker, "The "enrichment meter"–a simple method for measuring isotopic enrichment," Los Alamos National Laboratory, Tech. Rep. LA-4605-MS, 1970. [Online]. Available: https://www.osti.gov/servlets/purl/4036535

[93] G. Nelson and D. Reilly, *Passive Nondestructive Assay of Nuclear Materials*. Los Alamos National Lab, 1991, ch. Gamma-Ray Interactions with Matter.

[94] P. Mortreau and R. Berndt, "Determination of the uranium enrichment with the naigem code," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 530, no. 3, pp. 559 – 567, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168900204010885

[95] J. Sprinkle, A. Christiansen, R. Cole, M. Collins, S.-T. Hsue, P. Knepper, T. McKown, and R. Siebelist, "Low-resolution gamma-ray measurements of uranium enrichment," *Applied Radiation and Isotopes*, vol. 48, no. 10, pp. 1525 – 1528, 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0969804397001516

[96] R. Gunnink, W. Ruhter, P. Miller, J. Goerten, M. Swinhoe, H. Wagner, J. Verplancke, M. Bickel, and S. Abousahl, "MGAU: A new analysis code for measuring ^{235}U enrichments in arbitrary samples," *INIS*, vol. 25, 1 1994.

[97] L. Hiller, T. Gosnell, J. Gronberg, and D. Wright, "Calculating gamma-ray signatures from aged mixtures of heavy nuclides," in *2007 IEEE Nuclear Science Symposium Conference Record*, vol. 2, Oct 2007, pp. 1138–1142.

[98] A. J Peurrung, "Predicting 232U content in uranium," *PNNL-12075*, 01 1999. [Online]. Available: https://doi.org/10.2172/2657

[99] R. Rothe, "Extrapolated experimental critical parameters of unreflected and steel-reflected massive enriched uranium metal spherical and hemispherical assemblies," *INEEL/EXT-97-01401*, Dec 1997. [Online]. Available: https://www.osti.gov/servlets/purl/658339

[100] J. Stinnett, personal communication.